ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Aplicación de Asistente Virtual para la Búsqueda de Aparcamiento en Ciudades Inteligentes

(On the use of Virtual Assistant for Smart Cities Parking Service Deployment)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Jorge Calleja Lecanda

Julio - 2021

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Jorge Calleja Lecanda
Director del TFG: Jorge Lanza Calderón
Título: "Aplicación de Asistente Virtual para la Búsqueda de Aparcamiento en Ciudades Inteligentes"
Title: "On the use of Virtual Assistant for Smart Cities Parking Service Deployment"

Presentado a examen el día: 30 de julio de 2021

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Francisco Javier Madruga Saavedra
Secretario (Apellidos, Nombre): Luis Francisco Diez Fernández
Vocal (Apellidos, Nombre): Jorge Lanza Calderón

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

Agradecimientos

Estos años en la Universidad han sido un período de aprendizaje constante, no solo a nivel técnico, sino también a nivel personal. Y es por eso por lo que me gustaría agradecer a todas aquellas personas que han contribuido en este proceso.

En primer lugar, me gustaría agradecer a todos los profesores del Grado, y en especial a mi tutor, Jorge Lanza, por compartir sus conocimientos, por guiarme y ayudarme siempre que lo he necesitado, sin ellos no hubiera sido posible.

En segundo lugar, no podía dejar sin mencionar a Raúl Cabria y Ángel Herrero, compañeros y amigos tanto dentro como fuera de la Universidad. A mi grupo de amigos, Pablo, Víctor y Adrián, apoyo fundamental.

Finalmente, a toda mi familia, en especial a mis padres y hermanos por la confianza que han depositado en mí.

Resumen

En los últimos años, el desarrollo tecnológico ha permitido desplegar en el ámbito de las ciudades soluciones más eficientes, seguras y ágiles para sus ciudadanos, relacionadas con la salud pública, el medio ambiente, la movilidad o la seguridad física y digital, dando lugar a las denominadas Ciudades Inteligentes. Estas nuevas tecnologías contribuyen a incrementar la participación ciudadana a través de una gestión eficiente de los recursos, así como como al aumento de la calidad de vida de las personas.

Las Ciudades Inteligentes se apoyan en infraestructuras IoT compuestas por una red de sensores capaces de medir distintos parámetros y fenómenos. Esta información permite en la mejorar los servicios existentes y generar nuevos. Entre los datos adquiridos se encuentran aquellos que describen el estado del tráfico o la disponibilidad de aparcamiento. Es en este ámbito, la gestión del tráfico, donde las ciudades está poniendo muchos esfuerzos para entre otros disminuir el grado de polución que los desplazamientos generan.

Adicionalmente, la forma de interactuar con la información por parte de los usuarios finales está cambiando. Del paradigma usual a través de interfaces visuales, cada vez están adquiriendo más importancia los asistentes virtuales mediante voz, que proporcionan un interfaz de uso menos intrusivo, algo que es de especial relevancia durante la conducción.

El objetivo de este proyecto es evaluar y desarrollar una aplicación de asistente virtual para facilitar el acceso a la información de tráfico y aparcamiento disponible a través de las plataformas de gestión desplegadas en las ciudades inteligentes, y habilitar su uso para la mejora y eficiencia del tráfico en la ciudad.

Abstract

In recent years, technological development has made possible to deploy more efficient, secure and agile solutions in the field of cities for their citizens, related to public health, the environment, mobility or physical and digital security, giving rise to the so-called Smart Cities. These new technologies contribute to increase citizen participation through an efficient management of resources, as well as to increase people's quality of life.

Smart Cities are based on IoT infrastructures composed of a network of sensors capable of measuring different parameters and phenomena. This information allows existing services to be improved and new ones to be generated. Among the data acquired are those describing the state of traffic or the availability of parking. It is in this area, traffic management, where cities are putting a lot of effort in order to, among other things, reduce the degree of pollution that travel generates.

In addition, the way users interact with information is changing. From the usual paradigm of visual interfaces, voice-based virtual assistants are becoming increasingly important, providing a less intrusive user interface, which is of particular relevance when driving.

The aim of this project is to evaluate and develop a virtual assistant application to facilitate access to traffic and parking information available through the management platforms deployed in smart cities, and to enable its use for the improvement and efficiency of traffic in the city.

Índice general

Índice de figuras

Lista de acrónimos

ASK	Alexa Skills Kit
ADC	Alexa Developer Console
API	Application Programming Interface
APL	Alexa Presentation Language
AWS	Amazon Web Services
GE	Generic Enablers
NLG	Natural Language Generation
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IoT	Internet of Things
JSON	JavaScript Object Notation
NGSI	Next. Generation Service Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
OCB	Orion Context Broker
ONU	Organización de las Naciones Unidas
REST	REpresentational State Transfer
RPA	Robotic Process Automation
SDK	Software Development Kit
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TIC	Tecnologías de la Información y la Comunicación
TLS	Transport Layer Security
URL	Uniform Resource Locator

1 Introducción

1.1 Motivación

Las Ciudades Inteligentes o *Smart Cities* son ciudades basadas en el desarrollo urbano sostenible, que mediante la innovación y las Tecnologías de la Información y la Comunicación (TIC) buscan una mejor gestión y prestación de sus servicios. Se sustentan en plataformas que están compuestas por software y hardware que permiten el intercambio de información entre los distintos sistemas que conforman la ciudad inteligente. Gracias al despliegue de infraestructuras de la Internet de las Cosas (IoT, *Internet of Things*), esta información se obtiene, por ejemplo, de sensores habilitados por toda la ciudad que determinan una serie de parámetros en tiempo real como el estado del tráfico, la temperatura o la contaminación atmosférica [1]. Toda esta información recogida se pone a disposición de la ciudadanía, aunque gran parte no sea consciente de ello, directamente a través de diversas aplicaciones o indirectamente a través de la mejora de los servicios de la ciudad.

En la actualidad, el paradigma de ciudad inteligente está en una fase de consolidación, tras su eclosión durante la última década. En este sentido, existe una gran heterogeneidad de soluciones y plataformas, tanto orientadas a la gestión de las propias infraestructuras IoT de las ciudades como dirigidas al desarrollo de servicios para la ciudad y el ciudadano. Esto está en parte motivado por una expansión que ha hecho uso soluciones particulares y verticales enfocadas a entornos específicos en lugar de haber seguido un planteamiento transversal empleando soluciones estándar.

Desde la fundación FIWARE [2] se busca solventar esta problemática y ofrecer una plataforma estándar y abierta que facilite el desarrollo homogéneo de las ciudades inteligentes y los servicios disponibles en ellas. Así define tecnologías para la gestión, publicación y acceso a información de contexto recolectada de los dispositivos IoT presentes en la ciudad. Adicionalmente define un conjunto de modelos de datos que trata de estandarizar la forma en la que dicha información se representa.

Por otro lado, en los últimos años, están ganando importancia dentro del contexto de la inteligencia artificial y el acceso a la información, los denominados asistentes virtuales, ya que se caracterizan por facilitar la interacción entre los usuarios y los datos. Son capaces de encontrar y acceder a los datos de manera rápida y eficaz mediante el lenguaje natural [3].

Un aspecto fundamental de los asistentes virtuales [4] es el procesamiento del lenguaje natural, una combinación de la inteligencia artificial y la lingüística que pretende simular la capacidad humana para comprender y generar el lenguaje. Dentro de los sistemas conversacionales se ha producido una gran evolución con respecto al campo del reconocimiento automático del habla. Se ha pasado de reconocer palabras aisladas a frases completas. Esto se debe en gran medida a los avances en la definición de gramáticas, reconocimiento de diálogos, reconocimiento de entidades nombradas, etc. Además, las aproximaciones basadas en el aprendizaje automático a partir de los datos han mejorado también notablemente su rendimiento en comparación con los enfoques tradicionales basados en la definición manual de reglas.

A la vista de lo anterior, se identifica la viabilidad de modificar la forma en la que los ciudadanos pueden interactuar con la ciudad empleando interfaces menos invasivas en la mayoría de las situaciones y principalmente durante los traslados en vehículo propio. Por esto motivo, y a modo ejemplificativo, se plantea particularizar el caso de uso al entorno de conducción y. más concretamente, a la búsqueda de aparcamiento, una de las tareas más habituales en el día a día. La relación entre los nuevos avances tecnológicos y la conducción puede llegar a ser complicada. Aunque, las innovaciones tecnológicas y digitales pueden favorecer a la experiencia de ponerse al volante, en otros casos, su uso incrementa las posibles distracciones de los conductores y, por lo tanto, los riesgos de sufrir un accidente. Estos sistemas conllevan en muchas ocasiones retirar las manos del volante, apartar la vista de la carretera y estar más pendientes de estos dispositivos que de la conducción.

Por otro lado, gracias a la información disponible en la plataforma IoT de la ciudad se pretende solucionar los problemas relacionados con la búsqueda de aparcamiento. Independientemente del motivo por el que viajemos, todos en algún momento hemos sentido angustia o estrés a la hora de buscar donde aparcar. Si además se considera que las ciudades están en constante crecimiento, este problema se agrava, lo que con conlleva una mayor pérdida de tiempo y dinero.

1.2 Objetivos

El objetivo que se plantea en el proyecto es el desarrollo de una solución que haciendo uso de asistentes de voz virtuales, como el ofrecido por Amazon a través de su plataforma Alexa, se pueda homogeneizar el acceso a la información disponible a través de infraestructuras IoT. Se particularizará el desarrollo para el caso de uso de la búsqueda de aparcamiento en una ruta preestablecida en el ámbito de la ciudad de Santander dentro del marco de las soluciones IoT de ciudad inteligente que dispone.

Para completar este objetivo, surgen otros objetivos secundarios de carácter más técnico:

- Analizar y comprender el funcionamiento de la plataforma FIWARE, como plataforma de gestión de información de contexto.
- Adquirir nuevos conocimientos sobre los asistentes virtuales, en concreto sobre Alexa, el asistente virtual desarrollado por Amazon.
- Aprovechar las posibilidades que nos proporcionan los servicios web para el acceso de funcionalidades remotas.
- Implementar una skill o aplicación Alexa eliminando dentro de lo posible la dependencia con terceros.

1.3 Estructura de la memoria

Esta memoria se estructura en seis capítulos en los que se describe el trabajo realizado para lograr los objetivos fijados.

Tras realizar una puesta de contexto y explicar los motivos y objetivos de este proyecto en el primer capítulo, en el segundo capítulo se analiza el concepto de Ciudad Inteligente

(*Smart City*) y los principios en que se fundamenta. Se estudia la IoT como elemento habilitador de las mismas, gracias al cual se adquiere información contextual de la ciudad a partir de la que se generan los servicios de valor añadido que se ofrecen a los ciudadanos. En este entorno, se profundiza en el funcionamiento de FIWARE, plataforma para el desarrollo y despliegue de aplicaciones en la Internet del Futuro.

El tercer capítulo se centra en la descripción de la tecnología vinculada a los asistentes virtuales y cómo éstos, gracias a su sencillez, se han convertido en uno de los medios preferentes de interacción del usuario con el entorno.

El cuarto capítulo se centra en el estudio y puesta en práctica de los conceptos necesarios comunicarse con Alexa. Como resultado se plantea la arquitectura de la solución que da respuesta a las necesidades de este proyecto.

En el quinto capítulo, tras comprender las características de Alexa, se aborda el diseño, configuración y desarrollo de la propuesta del proyecto, integrando tanto el interfaz de interacción con la skill como el soporte de comunicación con la plataforma IoT.

Por último, en el sexto capítulo, se resumen las conclusiones del trabajo realizado y se extraen posibles líneas de trabajo futuro.

2 Ciudades inteligentes

Según la Organización de Naciones Unidas (ONU), el 70% de los seres humanos vivirá en centros urbanos para 2050 [5]. Este dato representa la importancia que están ejerciendo las ciudades como foco de atracción, lo cual plantea grandes desafíos a nivel demográfico, ecológico, social y económico. Por tanto, es necesario crear nuevos modelos de habitabilidad para satisfacer el aumento de la demanda de necesidades y servicios por parte de la población de las ciudades. Sin embargo, no todas las ciudades están preparadas para afrontar estos retos de la misma manera. No obstante, algunas han decidido aprovechar las herramientas que ofrecen la sociedad de la información y las telecomunicaciones para adaptar sus infraestructuras y servicios y lograr mejorar la gestión comunitaria de los recursos. Surgen así las denominadas ciudades inteligentes.

Aunque hoy en día todos estamos familiarizados con el término Ciudad Inteligente, también conocido por su acepción anglosajona de *Smart City*, lo cierto es que existe una gran diversidad de formas de definir el concepto tras ellas. Tomando como ejemplo la definición incluida en el Plan Nacional de Ciudades Inteligentes, liderado por la Secretaría de Estado de Telecomunicaciones en colaboración con el sector privado [6], una *"Ciudad inteligente (Smart City) es la visión holística de una ciudad que aplica las TIC para la mejora de la calidad de vida y la accesibilidad de sus habitantes y asegura un desarrollo sostenible económico, social y ambiental en mejora permanente. Una ciudad inteligente permite a los ciudadanos interactuar con ella de forma multidisciplinar y se adapta en tiempo real a sus necesidades, de forma eficiente en calidad y costes, ofreciendo datos abiertos, soluciones y servicios orientados a los ciudadanos como personas, para resolver los efectos del crecimiento de las ciudades, en ámbitos públicos y privados, a través de la integración innovadora de infraestructuras con sistemas de gestión inteligente."*

Parece lógico que, de la anterior descripción, las ciudades inteligentes se consideren uno de los instrumentos más potentes en políticas públicas a aplicar en las ciudades en un futuro próximo. No solo permitirán aplicar mejoras en la provisión servicios de valor añadido a los ciudadanos, sino que contribuirán a un desarrollo sostenible económico y social en las próximas décadas.

En este sentido, el despliegue de ciudades inteligente va a favorecer la gestión automática y eficiente de las infraestructuras urbanas, lo que conlleva, por un lado, la reducción del gasto, y, por otro, la mejora en sí de los propios servicios. Por tanto, si estas infraestructuras son desarrolladas de una forma flexible y bajo una aproximación holística que considere las necesidades de todos, todo el ecosistema podrá ser utilizado en el futuro para proporcionar servicios avanzados, que probablemente ni se consideraban en un principio.

Desde un enfoque más tecnológico, el concepto Ciudad Inteligente y el de IoT son dos conceptos que van de la mano. Las ciudades inteligentes constituyen un entorno urbano integrado e inteligente donde las soluciones de IoT se utilizan para interconectar, interactuar, controlar y proporcionar información sobre la multitud de sistemas, hoy en día fragmentados, dentro de las ciudades.

2.1 Internet de las Cosas

En la actualidad, las capacidades de comunicación se extienden a casi cualquier dispositivo que se pueda imaginar. La IoT se entiende como la red que conforman todos estos dispositivos y a través de los cuales se puede tanto adquirir información de estos, como actuar sobre ellos para que realicen operaciones. Todo esto es viable gracias a que los dispositivos, además de estar equipados con elementos de comunicación, normalmente incluyen alguna clase de inteligencia. Por lo tanto, se puede considerar que la IoT es una evolución de la tradicional Internet hacia un entorno más global con una mayor interconectividad, una mejor percepción de la información y servicios inteligentes más completos.

La IoT es una tecnología interdisciplinaria, que abarca múltiples áreas como la informática, las comunicaciones o la microelectrónica. No obstante, para una mayor simplicidad, la arquitectura de sistemas de IoT se puede dividir en cuatro capas, mostradas en la Figura 2.1:

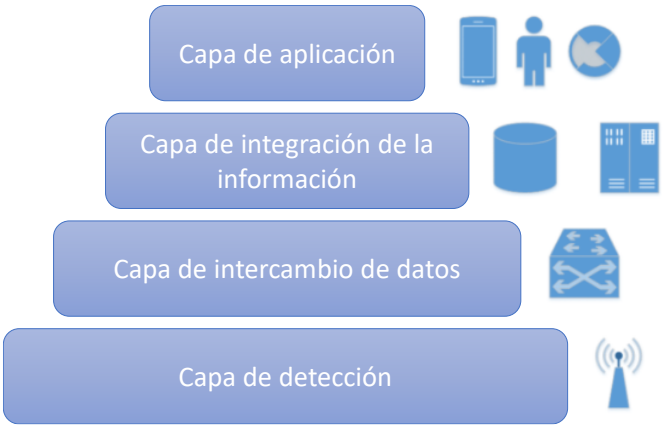

Figura 2.1: Arquitectura de la IoT

- La capa de detección es la capa física. Cuenta con sensores o actuadores para detectar y recoger información. Esta capa además de detectar determinados parámetros físicos en el entorno, es capaz de identificar otros objetos inteligentes en el mismo.
- La capa de intercambio de datos se encarga de la transmisión. Actúa como un puente transparente que transporta los datos recogidos por los dispositivos IoT. También conecta los dispositivos de la red y las redes entre sí.
- La capa de integración de la información recoge la información y datos de la capa subyacente y los procesa y filtra para entregar conocimiento útil a la capa de aplicación.
- La capa de aplicación utiliza los datos recolectados para proporcionar servicios a los usuarios. La información proporcionada por las infraestructuras IoT se pone a disposición de usuarios y proveedores de servicios. Haciendo usos de interfaces estandarizados se diseñan servicios innovadores orientados a los consumidores finales.

En general, una aplicación basada en la IoT requiere reunir los datos recogidos a través de los dispositivos de detección y procesarlos mediante diferentes algoritmos. Luego, se puede acceder a la información procesada a través de Internet. Desde el punto de vista de

la red se podría considerar otra estructura distinta basada en la agrupación de dispositivos o sensores (*clustering*), similar a la que se representa en la Figura 2.2.

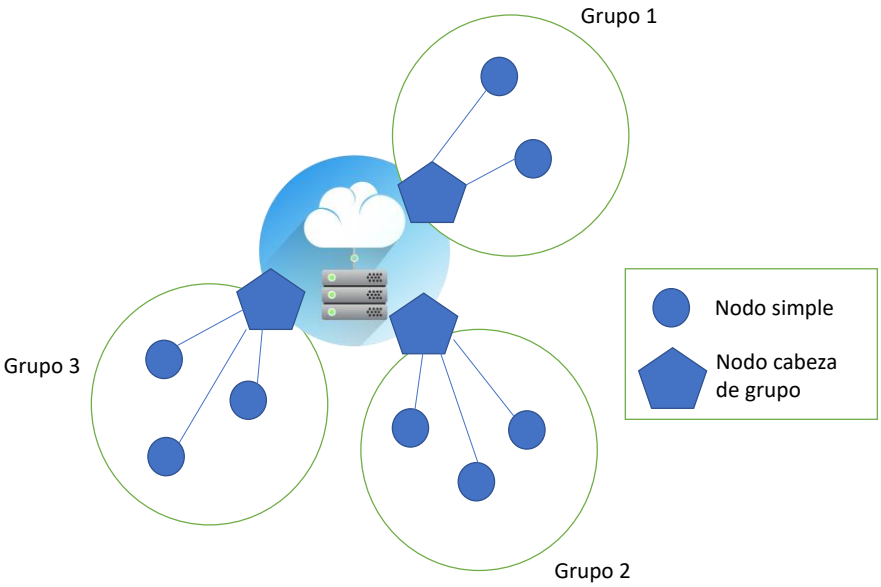

Figura 2.2: Estructura de la IoT basada en clustering.

El *clustering* facilita la tarea de acceder a la información con un número mínimo de comunicaciones dentro de una red y permite enviar esta información para su procesamiento posterior. En la agrupación, cada grupo elige un nodo como cabeza de grupo mediante el cual se lleva a cabo la comunicación con otros grupos de la red. Además, este nodo principal ayuda a gestionar y a agregar los datos adquiridos por diferentes nodos en una red para proporcionar los distintos servicios dentro de la plataforma de la IoT.

2.2 Plataformas de gestión de la IoT

Una plataforma IoT [7] engloba un ecosistema de dispositivos inteligentes interconectados, a partir de la cual extraer información contextual del entorno en el que se despliega. Debe contar con las características mostradas en la Figura 2.3.

Los dispositivos que estén conectados a la red deben funcionar de forma eficiente sin importar desde donde tenga lugar la conexión. Además, estas plataformas cuentan con una base de datos, no solo escalable en función de los requerimientos, sino que también está conectada a la nube.

La plataforma también se encarga de que los datos se adecuen a los patrones establecidos, además de a normas que permitan conocer el estado del sistema en todo momento. En caso de querer acceso a más información, existe disponible una función avanzada de análisis. A esto hay que añadir otras funciones como, por ejemplo, una interfaz de usuario atractiva que permite disponer de la información de manera visual a través de gráficos. Además, las empresas u organizaciones que quieran beneficiarse de estos sistemas pueden potenciarlos añadiendo, por ejemplo, aplicaciones que permitan dotar a la plataforma de más funcionalidades.

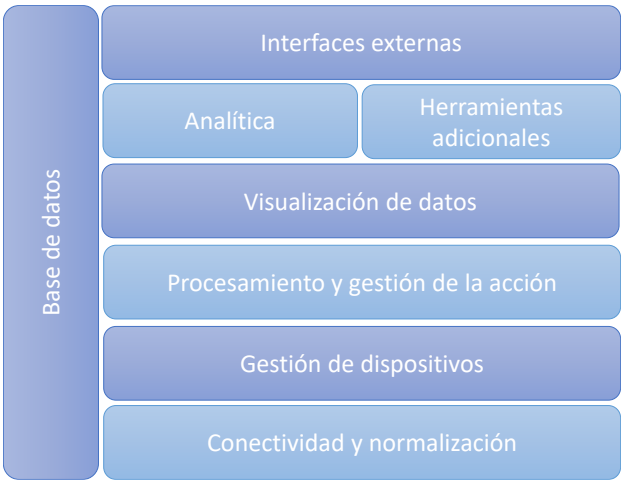

Figura 2.3: Características de una plataforma IoT.

Hay que destacar que estas soluciones son populares por su versatilidad, pero es necesario cierto control para asegurar la transmisión de información de forma confiable. En este sentido, es primordial que los encargados del área de la seguridad puedan verificar su correcto funcionamiento.

Dentro del marco europeo se puede encontrar algún proyecto dedicado a la creación de este tipo de plataformas y servicios comunes para el desarrollo de las ciudades del futuro, como SynchroniCity, OneM2M o FIWARE.

El proyecto SynchorniCity [8] establece una arquitectura de referencia para las ciudades inteligentes. Esta incluye herramientas para la creación y la integración de plataformas heredadas y dispositivos IoT para servicios urbanos. Forman parte de este proyecto, ciudades como Helsinki, Manchester, Milán, Oporto o Santander.

Por su parte OneM2M, [9] define una tecnología de middleware común en una capa horizontal entre los dispositivos y las redes de comunicaciones y las aplicaciones IoT. Esto estandariza los enlaces entre los dispositivos conectados, las pasarelas, las redes de comunicaciones y la infraestructura de la nube.

Finalmente, FIWARE ofrece un marco para el desarrollo de aplicaciones inteligentes en el entorno del Internet del Futuro. Proporciona la infraestructura necesaria para la prestación de nuevos servicios en el ámbito de las ciudades inteligentes. Algunos ejemplos de ciudades que cuenta con acceso a esta plataforma en España son Málaga o Santander. Dado que este trabajo se basa en esta plataforma, se detalla en profundidad sus características en el siguiente apartado.

2.3 FIWARE

FIWARE [2] es una iniciativa creada para proporcionar una plataforma y un conjunto de API para el impulso del desarrollo de soluciones inteligentes en el entorno de Internet del Futuro. Estas soluciones se caracterizan por recoger datos del entorno de distintos puntos finales como los usuarios, sensores o aplicaciones móviles.

De manera general la plataforma FIWARE proporciona las bases de la infraestructura de las ciudades inteligentes a través de las siguientes funcionalidades:

- Gestión de la información de contexto: dispone de los elementos necesarios para almacenar, acceder, procesar y analizar los datos como parte de una aplicación.
- Servicios de Internet de las Cosas: proporciona herramientas para configurar la red de sensores y enrutar la información de contexto.
- Hosting Cloud: permite gestionar los servicios a través de tecnología basada en la nube.
- Middleware avanzado para el intercambio de información entre agentes intermedios y aplicaciones.

La plataforma FIWARE ofrece capacidades de código abierto disponibles en la nube, además de un conjunto de herramientas y librerías de valor añadido, denominados Generic Enablers (GE). En la Figura 2.4 se muestra cómo los distintos GE de FIWARE se combinan para construir una arquitectura que resuelve las necesidades de una ciudad inteligente.

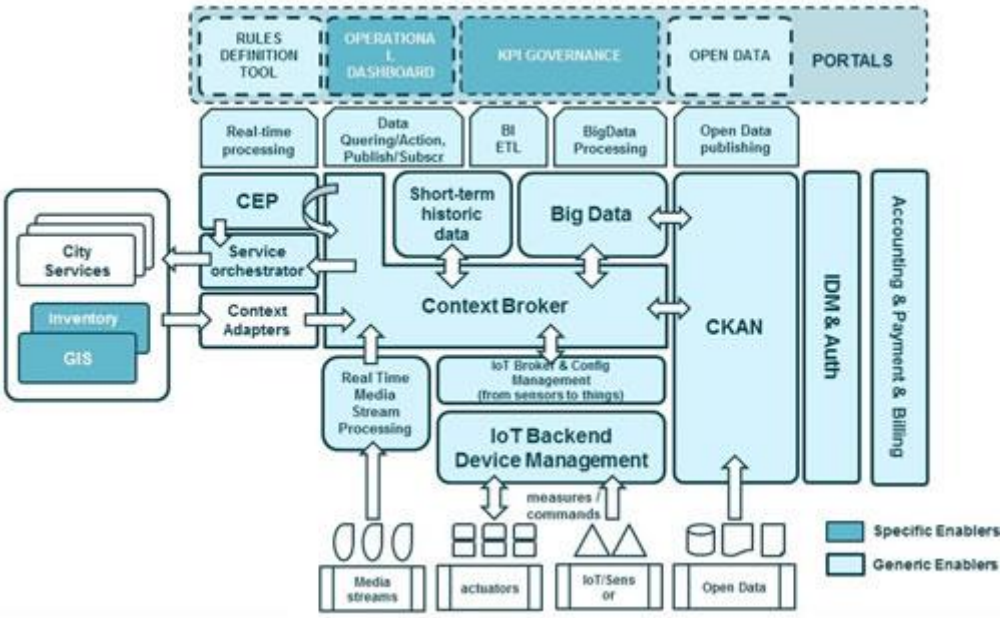

Figura 2.4: Arquitectura para Smart Cities FIWARE.

Se puede observar que el elemento central es el Orion Context Broker (OCB). Se trata del único componente obligatorio de cualquier solución desarrollada a partir de FIWARE, ya que se encarga de gestionar, consultar y actualizar la información de contexto.

Gracia al OCB se puede publicar la información de contexto recolectada por entidades como por ejemplo sensores. De esta forma la información es accesible por aquellos usuarios o proveedores de servicio interesados en procesar la información. Un ejemplo es la aplicación que se plantea en el ámbito de este trabajo que enriquece la interfaz de acceso a la información de los sensores de proximidad de los aparcamientos.

El OCB implementa un servicio basado en el modelo de información NGSI [10]. Este modelo permite realizar ciertas operaciones sobre la información de contexto como por ejemplo actualizar o consultar determinados datos, subscribirse a notificaciones, etc. Este servicio se lleva a cabo sobre la API RESTful NGSI v2.

La comunicación entre los distintos elementos de la arquitectura se sustenta en que en la información de contexto se encuentra estructurada a través de los elementos de contexto. Esta estructura está compuesta por distintos atributos. Por lo general, un elemento de contexto contiene un identificador (EntityId) y un tipo (EntityType) que identifica exclusivamente a una entidad. Además, pueden existir metadatos, vinculados a los atributos de un elemento de contexto como se muestra en la Figura 2.5.

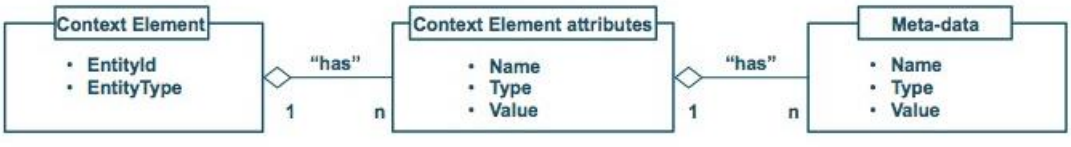

Figura 2.5: Estructura de los elementos de contexto.

Con objeto de homogeneizar el modelo de datos y estandarizar su uso, se considera un modelo de datos predefinido en función del tipo de elemento de contexto. En este sentido, se definen un conjunto de atributos específicos en función del entorno de aplicación. Se puede encontrar una amplia variedad de modelos de datos incluyendo entidades que gestionan datos medioambientales, alumbrado público, puntos de interés, gestión de residuos, etc.

A modo de ejemplo, vamos a profundizar en los de tipo de entidad "WeatherObserved", vinculada a información medioambiental y cuya representación se incluye en la Figura 2.6.


```
{
  "id": "Spain-WeatherObserved-Cantabria-2016-11-
30T07:00:00.00Z",
  "type": "WeatherObserved",
  "dateObserved": {
    "type": "DateTime",
    "value": "2016-11-30T07:00:00.00Z"
  },
  "illuminance": {
    "value": 1000
  },
  "temperature": {
    "value": 15.1
  },
  "precipitation": {
    "value": 0
  },
  "atmosphericPressure": {
    "value": 938.9
  },
  "pressureTendency": {
    "value": 0.5
  },
"windSpeed": {
    "value": 2
  },
  "location": {
    "type": "geo:json",
    "value": {
      "type": "Point",
      "coordinates": [-4.051096, 43.345322]
    }
  },
  "windDirection": {
    "value": -45
  },
  "relativeHumidity": {
    "value": 1
  }
}
```


Figura 2.6: JSON del modelo de datos "WeatherObserved".

De arriba a abajo se pueden observar en un principio el identificador único y el tipo de la entidad propio del modelo de datos de NGSI, así como la fecha y la hora desde la última actualización de información. Algunos de los parámetros medidos que están disponibles dentro de este tipo de entidad son: la humedad, la temperatura, la presión atmosférica, la velocidad o dirección del viento, entre otros.

No obstante, para este proyecto son objeto de interés las entidades de tipo "ParkingSpot", encargadas de controlar el estado de las plazas de aparcamiento, cuyo modelo se describirá más adelante.

Esta iniciativa ha sido acogida por ciudades de toda Europa como Torino, Lisboa, Ámsterdam o España, entre otras. En España destacan Santander, Málaga, Barcelona o Sevilla.

En el caso de Santander, la plataforma se presentó en 2013 [11], convirtiéndose en plataforma clave para la gestión de la información recogida por los dispositivos de la IoT, dispersos por toda la ciudad, como sensores embebidos en parques, aparcamientos o autobuses.

3 Asistentes virtuales

En el pasado cuando se hablaba de un fácil acceso a la información, lo expertos buscaban la forma de mejorar la navegación minimizando el número de clics. Así, por ejemplo, para no usar navegación visual se empezaron a plantear sistemas de bots conversacionales que se implementaban mediante teclado. Estas aplicaciones surgidas en los años 60 tenían por función simular una conversación mediante respuestas automáticas de texto a problemas comunes o conocidos. Desde su origen han experimentado una gran evolución, y gracias a la inteligencia artificial, la interacción hombre-máquina se puede llevar a cabo mediante el uso de la voz. Se pueden considerar como la tecnología previa a los asistentes virtuales.

En la actualidad, se busca que el sistema sea capaz de reconocer las intenciones de los usuarios, identificar sus sentimientos y dar recomendaciones en función de sus gustos y necesidades. Estos sistemas incorporan un módulo de inteligencia artificial que responde a las peticiones del usuario a través de una interfaz, en donde la voz, la síntesis, el reconocimiento y procesamiento de lenguaje natural es fundamental.

La evolución de los sistemas de procesamiento de lenguaje natural (NLP, *Natural Language Processing*) y generación de lenguaje natural (NLG, *Natural Language Generation*), unido al desarrollo de los teléfonos inteligentes en la sociedad, ha llevado a la aparición de los llamados asistentes virtuales.

Se puede definir un asistente virtual como un agente de software capaz de interactuar con el usuario a través de un lenguaje natural, identificando, procesando e interpretando la voz del usuario. Cabe mencionar que estos agentes están siendo utilizados cada vez más por las empresas como soporte postventa y atención al cliente, mejorando la experiencia del cliente con la empresa. Además, se pueden encontrar tanto en altavoces inteligentes como en teléfonos de última generación o en aparatos electrónicos.

Gracias al reconocimiento y comprensión de lenguajes naturales en múltiples idiomas, están a disposición de cualquier usuario y permiten el acceso a servicios de valor añadido, aunque éste no posea conocimientos técnicos. Construyen sus respuestas a partir de información contextual actualizada en tiempo real o de sesiones anteriores. En este sentido emplean, para enriquecer el contenido de la respuesta, todo tipo de información disponible, como la ubicación del usuario, perfil de mismo, etc. La personalización y la calidad de las respuestas dependen de la cantidad, pero más aún de la calidad de los datos. En este sentido, se debe proceder al correcto tratamiento, limpieza, formateo de los datos disponibles, y armonizarlos con otros de fuentes externas. Para llevar a cabo esta labor, las técnicas de aprendizaje inteligente jugarán un papel relevante

Hoy en día, con la nueva generación de asistentes virtuales y su nueva interfaz, la cual también permite interactuar través de videos, imágenes, sentimientos y gestos del usuario, se facilitará un mayor grado de personalización al poder reconocer las emociones y el contexto del dialogo durante la conversación establecida.

Sin embargo, hay que tener en cuenta a la hora de implantar estos asistentes virtuales que son sumamente especializados en un tema de conocimiento concreto, limitando su actividad a aquellas relacionadas con su tema de conocimiento. Esto se debe a que el núcleo lleva un módulo cognitivo que va aprendiendo para ajustarlo a las necesidades

intrínsecas de su tema de conocimiento. Al tratar otros temas, tiene dificultades de aprendizaje, y no daría respuesta para a peticiones sobre otros temas.

3.1 Componentes de un asistente virtual

La arquitectura de los asistentes virtuales consiste en la integración de diferentes módulos ya empleados en aplicaciones independientes y que son incorporados en una única aplicación, incluyendo sus puntos fuertes. Un asistente virtual se compone de los siguientes elementos, Figura 3.1:

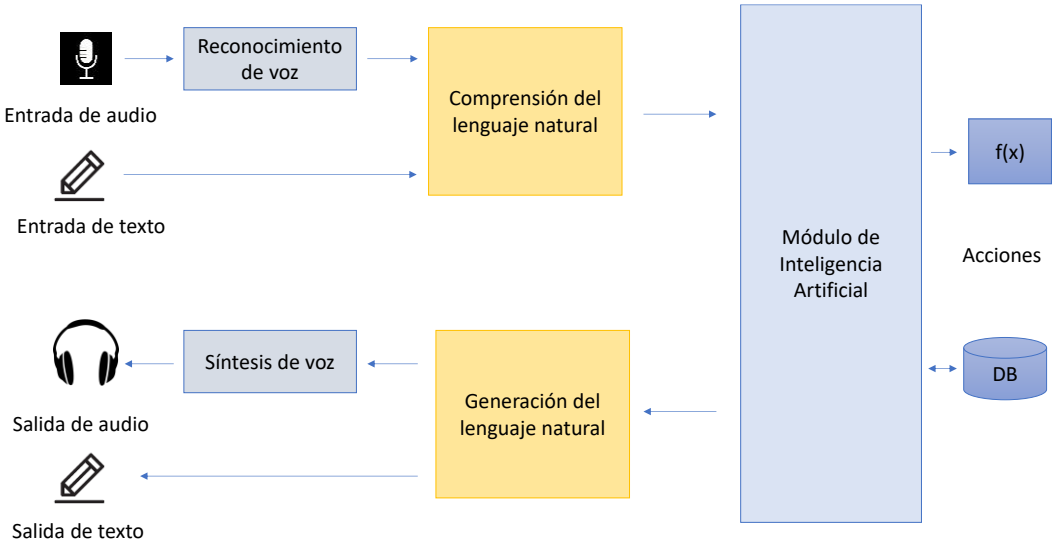

Figura 3.1: Diagrama de flujo de la información en un asistente virtual.

- Un módulo de conversación que permite la comunicación con el usuario bien sea texto a texto, voz a texto, texto a voz o voz a voz.
- Un módulo de NLP y un módulo de NLG que se encargan de la compresión del mensaje y de la intención del usuario, así como de la respuesta del asistente virtual hacia el usuario en el mismo idioma.

  Los primeros y sencillos asistentes virtuales establecían patrones predefinidos, que trataban de identificar en el mensaje que verbaliza el usuario. Por lo tanto, el sistema carecía de contexto. Por lo general, este tipo de NLP utiliza además un análisis sintáctico para evitar la sinonimia mediante expresiones regulares que establece reglas para relacionar términos, con lo que se consigue que aquellos similares se identifiquen bajo una misma acción.

  En los asistentes virtuales más complejos se busca una interpretación real del sentido de la pregunta, frase o conversación. En este caso se introducen relaciones semánticas mediante homonimias o arboles de diálogo, dando a la expresión un significado completo.
- Un módulo de inteligencia o cognitivo propio de las aplicaciones RPA (Robotic Process Automatic), que permiten desde devolver un simple mensaje al usuario hasta controlar las acciones de algún aparato electrónico. Está constituido por un motor de razonamiento capaz de procesar las instrucciones que le llegan a través del NLP. Trata cada funcionalidad como un componente

independiente, y gestiona las comunicaciones entre ellos a partir de una serie de eventos que se disparan ante una petición. Esta comunicación se realiza bajo un API REST que identifica el punto de acceso al servicio. La conexión a los servicios se produce mediante el protocolo HTTP (método GET o método POST).

Dentro de los sistemas de reconocimiento de voz, hay un aspecto crucial en su diseño, y es la elección del tipo de aprendizaje que se va a utilizar como fuente de conocimiento [12].

- Aprendizaje deductivo: este tipo de aprendizaje se basa en el conocimiento aprendido, es decir, en la transferencia de conocimientos por parte del ser humano hacia el sistema. Un ejemplo de este tipo de aprendizaje son los Sistemas Expertos, los cuales simulan el razonamiento humano tal y como lo haría un experto en un área de conocimiento.
- Aprendizaje inductivo: el sistema es capaz de conseguir por sí mismo los conocimientos necesarios para, a partir de ejemplos reales, realizar la tarea. Es el caso de las redes neuronales artificiales.

En la práctica, ningún sistema está basado únicamente en solo uno de los tipos de aprendizaje, se asume una metodología deductiva-inductiva en la que los aspectos generales se suministran deductivamente y la caracterización de la variabilidad inductivamente.

3.2 Soluciones populares

Los principales jugadores que están influyendo en el marco de los asistentes virtuales y fomentando su desarrollo son los gigantes tecnológicos: Google, Amazon, Microsoft y Apple. A continuación, se profundizará sobre cada uno de sus asistentes virtuales además de una breve mención a la solución de Samsung.

3.2.1 Alexa

Alexa [13] es el asistente virtual desarrollado por Amazon, que además lo incorpora en diferentes dispositivos físicos dedicados. En un principio solo estaba disponible en los altavoces inteligentes creados por Amazon, pero tras un periodo inicial de exclusividad Amazon abrió su SDK para que otros desarrolladores pudieran incorporarlo en sus plataformas. Desde entonces, el asistente está presente en una gran variedad de dispositivos como relojes, televisiones o incluso electrodomésticos.

Dentro de las funciones de Alexa se pueden encontrar todo tipo de comandos de voz, con los que se pueden realizar una amplia variedad de solicitudes. Algunos de los comandos disponibles incluyen acciones como pedir información sobre cualquier tipo de contenido, ya sea información meteorológica, datos específicos sobre personas y/o productos, etc. También permite llevar a cabo la configuración de alarmas o establecer recordatorios, entre otras muchas funcionalidades.

La biblioteca de funcionalidades implementadas por defecto en Alexa puede ampliarse a través de las Alexa Skills, consideradas como el equivalente a aplicaciones o funcionalidades de terceros. Actualmente existen aproximadamente más de 100.000

skills. En el capítulo 4, se detallará en profundidad las características y funcionamiento de las mismas.

3.2.2 Google Assistant

Google Assistant [14] es el asistente inteligente de Google. Se caracteriza por ser multiplataforma, gracias a su app propia cuenta con todas las funcionalidades tanto para iOS como para Android.

Google Assistant no solo es una extensión de la búsqueda de Google, sino que en realidad puede comportarse de la misma forma que lo haría Siri o Cortana. Por ejemplo, permite proporcionar información basada en el contexto del usuario, ya sea su localización, el momento del día, sus preferencias, etc. Así podrá notificar al usuario con noticias de interés basadas en su localización, en sus interés o historial de búsqueda. Su operativa se sustenta en el motor de búsqueda de Google.

3.2.3 Siri

Siri [15] es posiblemente el asistente virtual más conocido y empleado junto con Alexa. En un principio fue desarrollado como una aplicación independiente, pero en 2011 Apple lo incorporó de forma integral a sus sistemas operativos.

Siri fortalece las funciones relacionadas con el teléfono móvil y la comunicación en general. Puede leer notificaciones y correos, escribir mensajes e incluso publicaciones de Facebook y Twitter, ayudarnos con nuestra agenda, etc.

Una de las principales ventajas de Siri es la privacidad. Si bien, Amazon y Google registran todas las solicitudes que llegan al altavoz, para más tarde mostrarnos publicidad personalizada, Apple es más respetuosa en este aspecto con la privacidad y no asocia información con el perfil de la persona con la que Siri conversa. Para lograr esto, Siri verifica todas las solicitudes con un ID anónimo.

3.2.4 Cortana

Cortana [16] es el asistente virtual de Microsoft. Es un componente fundamental en Windows 10 y además se puede integrar en sistemas iOS, Android y Xbox mediante una aplicación.

Dentro de sus características que el sistema cuenta con una sección donde recopila información personal del usuario: lugares de interés, gustos, etc. Posteriormente, se puede modificar la información recogida y, según el momento, el asistente proporciona automáticamente consejos y sugerencias.

Microsoft ofrece, además de la interfaz de voz, interactuar mediante el teclado, lo que ayuda a la comprensión, en lugares donde la voz tendría problemas, como son entornos con mucho ruido. Además, esta interfaz adicional permite extender su aplicación al entorno de escritorio en portátiles u ordenadores de sobremesa.

En la actualidad, Microsoft ha abandonado el desarrollo de este asistente, debido a la fuerte competencia de los asistentes mencionados en las secciones anteriores. A partir de

Windows 11 ya no estará disponible, aunque continua como complemento en el paquete de servicios de Microsoft 365.

3.2.5 Bixby

Bixby [17] es el asistente virtual de Samsung, disponible únicamente en las últimas versiones de los dispositivos de la empresa.

Se diferencia del resto de los asistentes en que se orienta fundamentalmente a ofrecer un interfaz de interacción adicional con el dispositivo bajo el paradigma de que todo lo que se puede hacer tocando, se puede hacer hablando. Así, por ejemplo, además de las funciones más tradicionales de otros asistentes, Bixby permite podría rotar fotografías, aplicar filtros, ejecutar aplicaciones, etc.

4 Alexa Skills

En la sección anterior se introdujo el concepto de skill, en el cual basa su funcionamiento el asistente virtual de Amazon. A continuación, se profundizará en su definición y sus características, y se introducirán las metodologías que ayudan al desarrollo de skills como la objeto de este proyecto.

4.1 Conceptos generales

Las skills son funcionalidades activadas por voz que los desarrolladores pueden crear para el servicio de Alexa en la nube y al cual pueden acceder los dispositivos de Amazon, como el Echo. Por defecto, los dispositivos de la serie Echo de Amazon incorporan una serie de funcionalidades predefinidas, y es precisamente con estas skills con las que se pueden ampliar. De forma simplificada una skill se puede considerar como el equivalente a las aplicaciones tradicionales pero aplicadas al asistente de voz.

Cuando el usuario quiere acceder a las funcionalidades de una skill, debe primeramente despertar a Alexa mediante la palabra de activación, que recibe el nombre de *wake word.* Una vez pronunciada, Alexa entiende que se está interactuando con ella, y continúa escuchando el resto de la frase. En el momento que se ha obtenido su atención, se debe indicar la skill de todas las disponibles se quiere ejecutar, para lo cual se la identifica a través del *invocation name*. Es a partir de entonces cuando el dispositivo se comunica con el servicio de Alexa en la nube, donde se reconoce el discurso, se determina lo que el usuario quiere y se envía la correspondiente solicitud invocando a la skill para que satisfaga la petición. El establecimiento y mantenimiento de la conversación con la skill es posible gracias al *Automated Speech Recognition* (ASR) y al *Natural Language Understanding* (NLU). De una parte, el ASR se encarga de identificar las palabras que ha pronunciado una persona y de traducirlas a texto, mientras que de otra el NLU, entendido como un subconjunto de NLP, estructura la información procedente de la petición del usuario de una forma que una maquina pueda comprenderla y actuar sobre ella.

En este sentido, Alexa se comunica con la inteligencia de una skill a través de solicitudes y respuestas HTTP, en las que dentro de la solicitud POST se encapsula un cuerpo JSON que incluye los parámetros necesarios para que la skill comprenda la solicitud, realice su lógica y, por último, genere una respuesta. En la Figura 4.1 se muestra en el diagrama de flujo de la operativa descrita.

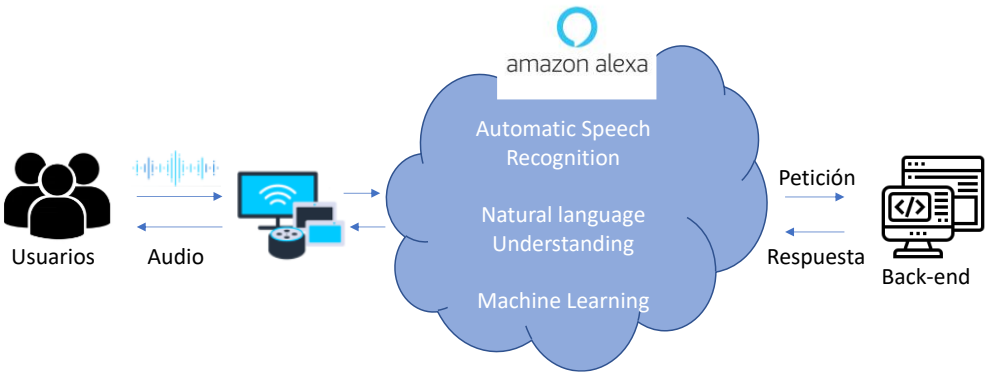

Figura 4.1: Diagrama de flujo del procesamiento de la voz.

En dicha figura, se distingue, por un lado, a los usuarios interactuando con los dispositivos que integran Alexa, y por otro, la operativa que recoge la lógica de la skill. Por lo tanto, además del back-end que desarrolle la inteligencia de la skill, es decir, el acceso a la información, el trabajo sobre los datos, etc., se requiere definir una interfaz de voz, como front-end, la cual determina lo que la skill puede comprender cuando el usuario habla. Ambas partes dan forma a una skill completa. En los próximos apartados se profundiza en ambos conceptos, explicando cada elemento del modelo de la interacción de voz, así la interfaz que debe implementar la lógica de la skill para que aceptar las peticiones.

4.1.1 Modelo de interacción de la voz

Cada skill tiene un modelo de interacción de voz que define las palabras y frases que los usuarios pueden emplear para controlar el comportamiento de una skill y que ejecute una determinada funcionalidad. A la hora de diseñar un modelo de interacción de voz, es necesario fijar una serie de elementos, como los *intents*, las *utterances*, los *slots* o el ya mencionado anteriormente, *invocation name.* A continuación, se detalla cada uno de ellos.

Una vez iniciada la interacción con la skill, el usuario debe pronunciar el *invocation name* para que el asistente Alexa pueda reconocer hacia qué skill redireccionar las peticiones del usuario.

El *invocation name* o nombre de invocación ha de cumplir una serie de condiciones. Debe constar de dos o más palabras, y sólo puede contener caracteres alfabéticos en minúscula, espacios entre palabras, o espacios y puntos utilizados en abreviaturas (por ejemplo, "a. b. c."). Además, no pueden contener ninguna de las frases de lanzamiento de la skill de Alexa, como "lanzar", "preguntar", "decir", "abrir", "cargar" o "empezar". Tampoco están permitidas las palabras "Alexa", "Amazon", "Echo", o "Skill" ni conectores como, por ejemplo, "por", "de", "en", "con", "para", "sobre", entre otros.

Previamente se ha mencionado que el *invocation name* debe constar de dos o más palabras, pero existen ciertos casos en los que podría estar formado por una única palabra, siempre y cuando sea única para su marca.

Además del *invocation name*, el usuario emplea los denominados *utterances* como metodología para establecer un diálogo con la skill y remitir información como una frase especifica o un dato. En general estas frases indican una acción concreta que el usuario quiere que la skill realice. Un ejemplo sería "Dime que películas se estrenan hoy". Cada

uno de los enunciados o utterances que se provean, se emplean como material de entrenamiento y aprendizaje para el asistente. Por tanto, no es necesario considerar todas las posibilidades que el usuario podría decir, pero sí tratar de maximizar la información proporcionada.

Cada conjunto de *utterances* se mapean sobre un *intent* especifico. Los *intent*s representan acciones que el usuario puede llevar a cabo con la skill y cada uno se corresponde con una funcionalidad concreta. En la Figura 4.2 se muestra cómo diferentes *utterances* se asocian a un mismo *intent*, permitiendo al usuario solicitar la misma acción de maneras distintas.

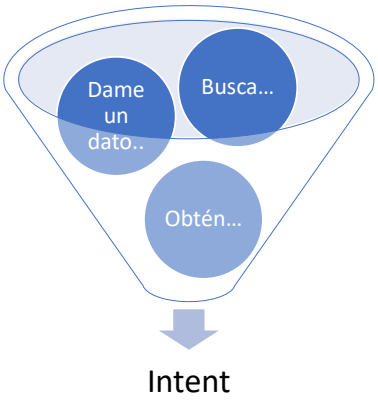

Figura 4.2: Mapeo de utterances.

Dentro de los tipos de *intents* que se pueden encontrar están los de tipo *Custom*, creados por el usuario y en los que se centra este proyecto, y los predefinidos, de tipo de *built-in*.

Los *intents* predefinidos o *built-in intents* son los siguientes:

- AMAZON.CancelIntent, cancela alguna interacción en proceso. Se puede utilizar en caso de introducir un valor equivocado en un *slot* antes de finalizar la interacción con el usuario. Los comandos o *utterances* que se suelen emplear son “cancela” u “olvídalo”.
- AMAZON.StopIntent, detiene por completo la skill. Algunos de los enunciados o frases que se suelen usar para este *intent* son “para” o “apaga”.
- AMAZON.HelpIntent, proporciona ayuda al usuario para usar correctamente la skill. Ofrece información sobre las capacidades de esta. Las frases o expresiones más comunes para este *intent* son “ayuda”, “ayúdame” o “necesito ayuda”.
- AMAZON.NavigateHomeIntent, equivale a un “regresa al inicio”. Está presente en los dispositivos con pantalla como los de la familia Echo, donde permite salir de la skill y devuelve a los usuarios a la pantalla de inicio. Los comandos más comunes son "Alexa, ve a casa" o "Alexa, ve a la pantalla de inicio".

Puesto que los mensajes que se remiten a la skill no tienen por qué ser estáticos, sino que el usuario puede incluir información variable, Alexa habilita un mecanismo para definir qué partes de cada *utterance* se corresponden con ese tipo de dato. Un *slot*, introducido anteriormente, recoge los valores de ciertos parámetros que la skill necesita

para cumplir su función. Estos argumentos, se pasan dentro de una *utterance*, tal y como se muestra en la Figura 4.3.

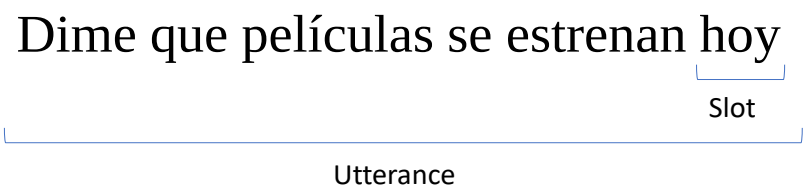

Figura 4.3: Ejemplo de utterances con slot.

Los slots se definen en función del tipo de dato que haga referencia. En algunos casos es necesarios definir *slots* propios, como por ejemplo un *slot* de tipo planeta, que recoja toda la casuística de nombres de planetas del sistema solar. No obstante, lo habitual es hacer uso de una lista de tipos de *slots* que ya vienen predefinidos como por ejemplo AMAZON.DATE, AMAZON.NUMBER, AMAZON.TIME, entre otros. El ejemplo mostrado en la Figura 4.3 incluye un tipo AMAZON.DATE, el cual convierte palabras que indican fechas ("hoy", "mañana" o "julio") en un formato de fecha (como "30-07-2021").

Una vez definidos estos elementos, Alexa genera un fichero JSON que contiene el cuerpo de la solicitud con los parámetros necesarios para que la skill funcione adecuadamente.

4.1.2 Lógica de la skill

Tras analizar y diseccionar el *utterance*, Alexa es capaz de generar una petición que incluya toda la información necesaria para que la lógica de la skill se ejecute y genere la correspondiente respuesta.

A la hora de implementar el servicio que acepte las solicitudes de Alexa, se ofrecen dos opciones: usar una función AWS Lambda o referenciar un servicio web externo. El primer caso ofrece una curva de aprendizaje más rápida, pues toda la ejecución y gestión de la skill se realiza bajo el ecosistema de Amazon. No obstante, el externalizar la lógica en un servidor externo permite una mayor independencia y únicamente requiere habilitar un servicio web que acepte las peticiones POST HTTP mediante las cuales Alexa transfiere la información.

A continuación, se describen ambas opciones, aunque en el ámbito de este trabajo se opta por la segunda opción, para eliminar dependencias con Amazon en la medida de lo posible.

4.1.2.1 Alojar una Skill como una función AWS Lambda

AWS Lambda es un servicio cloud que proporciona un medio para la ejecución de código sin la necesidad de gestionar servidores, permite crear una lógica de escalado basada en la carga de trabajo, es decir, solo se ejecuta cuando es necesario. Por lo tanto, para ejecutar código simplemente es necesario cargar el código y la función Lambda asigna de manera automática y precisa los recursos informáticos y ejecuta el código en función de la solicitud para cualquier escala de tráfico.

En general, alojar la skill en una función Lambda elimina la mayor parte de la complejidad a la hora de configurar un endpoint o servicio web propio, aportando las siguientes ventajas:

- No es necesario administrar o gestionar ninguno de los recursos del servicio.
- No necesita un certificado SSL.
- Como el acceso al servicio se controla mediante permisos dentro de AWS, no es necesario verificar si las peticiones proceden de Alexa.
- El código desarrollado solo se ejecuta cuando se necesita y se escala de la misma manera, por lo tanto, no hace falta suministrar recursos continuamente.
- La comunicación está encriptada con TLS.

4.1.2.2 Alojar una Skill como un Servicio Web

Otra posibilidad que se brinda a los desarrolladores, y la elegida en este proyecto, es alojar la skill creada como un servicio web externo, que acepte peticiones y envíe la respuesta al servicio de Alexa en la nube. El servicio web puede estar escrito en cualquier tipo de lenguaje, siempre y cuando, se cumpla con los siguientes requisitos:

- Ser accesible a través de Internet.
- Aceptar peticiones HTTP en el puerto 443.
- Soportar HTTP sobre SSL/TLS, utilizando un certificado de confianza.
- Verificar que las solicitudes entrantes provienen de Alexa.
- Adherirse a la interfaz del Alexa Skills Kit.

A la hora de establecer la confianza con el servicio web, éste debe de disponer de un certificado que cumpla con las condiciones establecidas por Amazon. Así se puede elegir entre tres opciones. La primera consiste en utilizar un certificado de confianza de Amazon, como por ejemplo alguno de los incluidos en la lista del Programa de Certificación CA de Mozilla [18]. La segunda opción se corresponde con la utilización de un certificado comodín, es decir, certificados con validez para varios subdominios. En cualquier caso, este certificado deberá estar firmado por una autoridad de certificación de confianza de Amazon, incluida en la lista previamente mencionada en la primera opción. La última opción consiste en utilizar un certificado autofirmado, que en general solo se utiliza con fines de prueba. Habrá que notificar sobre este certificado a Alexa, acción que se realiza al cargarlo en el repositorio habilitado en la consola de desarrollo del Alexa Skill Kit.

Otro de los aspectos más importante para proteger el servicio web de potenciales atacantes es verificar que las solicitudes entrantes son enviadas por Alexa. Las solicitudes que no provengan de Alexa deben ser rechazadas. Para evitar tener que verificar las peticiones manualmente, se puede desarrollar el servicio utilizando los SDK del Alexa Skills Kit (ASK SDK) para Node.js, Java o Python que incluyen API que realizan el proceso de forma automática.

4.2 Ejemplo práctico

Definida la operativa de Alexa, así como los conceptos del modelo de interacción de voz y lógica de la skill, se va a plantear un ejemplo sencillo para introducir a los desarrolladores en el entorno de programación. Este ejemplo se basará en un clásico de la programación, un "Hello World". Para este ejemplo se supone que la lógica de acceso al servicio web ya esté implementada, así como el código que devuelve un "Hola mundo". No obstante, en el capítulo 5, se explicará su implementación para nuestra skill objetivo.

El Alexa Skills Kit (ASK) es un entorno de desarrollo de software capaz de crear y administrar skills. Éste contiene un framework llamado Alexa Developer Console (ADC). Para poder acceder por primera vez a la herramienta es necesario crear una cuenta de Amazon como desarrollador. Una vez hecho se puede acceder a la pantalla de inicio mostrada en la Figura 4.4.

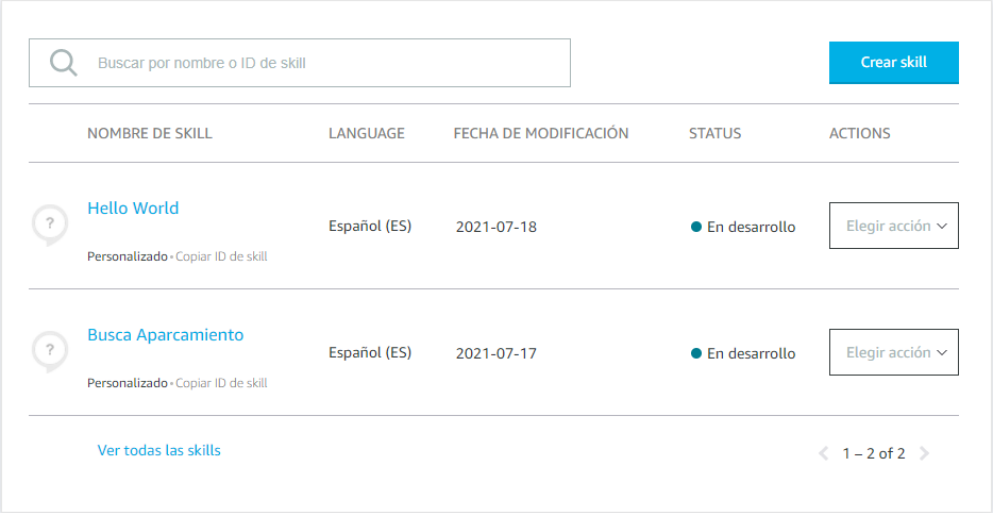

Figura 4.4: Interfaz de inicio de ADC.

Para iniciar el proceso de crear una skill solo es necesario elegir la opción "Create Skill" y elegir un nombre, un método de alojar la skill y una plantilla como base de su desarrollo. Una vez hecho esto, se cuenta con dos herramientas para poder trabajar en el desarrollo de las skills. La primera de ellas es el Build donde se puede construir todo el front-end. La segunda, Test, permite poner a prueba el funcionamiento de la skill. Existe una tercera herramienta Code, en la que se puede desarrollar el back-end siempre y cuando se elija AWS Lambda como método de desarrollo, alojamiento y ejecución de la lógica de la skill, pero no es el caso que aplica al desarrollo enmarcado en este trabajo.

Continuando con el ejemplo, en el Build se tiene acceso a todo lo necesario para construir la parte más cercana al usuario. La mayor parte del tiempo se pasa en esta sección, generando el modelo de interacción de voz.

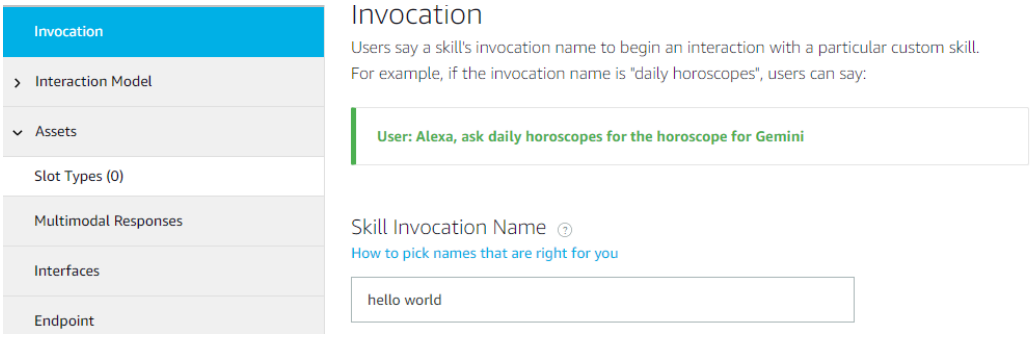

Figura 4.5: Invocation.

En la Figura 4.5 se muestra la sección "Invocation" donde se da un nombre de invocación, necesario para que Alexa pueda reconocer la skill y sea capaz de lanzarla. Para nuestro ejemplo se ha elegido como nombre de invocación el mismo nombre de la skill "hello world".

El siguiente paso sería establecer los *intents*, elemento fundamental sin el que la skill no tendría sentido, ya que los *intents* representan las acciones que los usuarios pueden realizar con ella.

La consola proporciona un apartado completo para el desarrollo de los *intents*. Por una parte, están los de tipo "Custom" creados por el usuario, y los de tipo de "required", que hacen referencia a los *built-in intents* que ya vienen predefinidos.

Ahora bien, dentro de los *intents* creados por el desarrollador de la skill, se va a proceder con el *intent* principal de Hello World, "HelloWorldIntent".

Figura 4.6: HelloWorldIntent.

En la Figura 4.6, se pueden observar las *utterances* para llamar al *intent* creado, "Hola", "como estás", "di hola mundo", "di hola" u "hola mundo", recordemos que las *utterances* se utilizaban para llamar una funcionalidad concreta. Por lo tanto, es importante tener en cuenta no repetir *utterances* en distintos *intents* para no interferir en el funcionamiento de cada uno. Las *utterances* pueden tener argumentos, los *slots*. En el caso que aplica, no están definidos en las expresiones ya que la skill solo responde con un "Hola Mundo" por lo que no necesita de ningún dato adicional. En el caso de que se necesitara definir un *slot*, se haría dentro de la propia *utterance*. De esta forma se crearía un *slot* sin un tipo asignado, el cual habría que definir a partir de una serie de opciones disponible o bien crear un nuevo tipo de slot, como se muestra en la Figura 4.7.

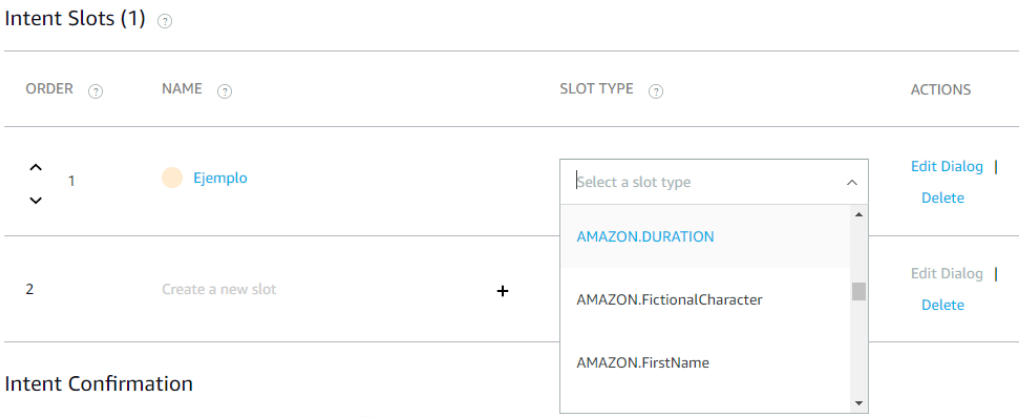

Figura 4.7: Ejemplo de creación de slots.

De esta manera se tiene una primera versión del modelo de interacción de voz listo. Solo queda validarlo y activarlo, tarea que se lleva a cabo mediante la opción de "Build Model". En la Figura 4.8 se muestra el fichero JSON resultado del análisis y disección de la información introducida a través de la consola. Este JSON se puede ver y modificar una vez generado en un editor que proporciona el propio entorno.

```
{
    "interactionModel": {
        "languageModel": {
            "invocationName": "hello world",
            "intents": [
                {
                    "name": "AMAZON.CancelIntent",
                    "samples": []
                },
                {
                    "name": "AMAZON.HelpIntent",
                    "samples": []
                },
                {
                    "name": "AMAZON.StopIntent",
                    "samples": []
                },
                {
                    "name": "HelloWorldIntent",
                    "slots": [],
                    "samples": [
                        "hola",
                        "como estás",
                        "di hola mundo",
                        "di hola",
                        "hola mundo"
                    ]
                }
            ],
            "types": []
        }
    }
}
```

Figura 4.8: Editor JSON.

Finalmente, en la sección de “Endpoint” de la consola, incluida en la Figura 4.9, se puede definir el método para alojar la skill. En este caso, se ha supuesto que la lógica de acceso al servicio ya ha sido creada, por lo que únicamente sería necesario introducir la URL donde se encuentra el servicio y elegir uno de los tres certificados aceptados por Alexa, ya explicados anteriormente.

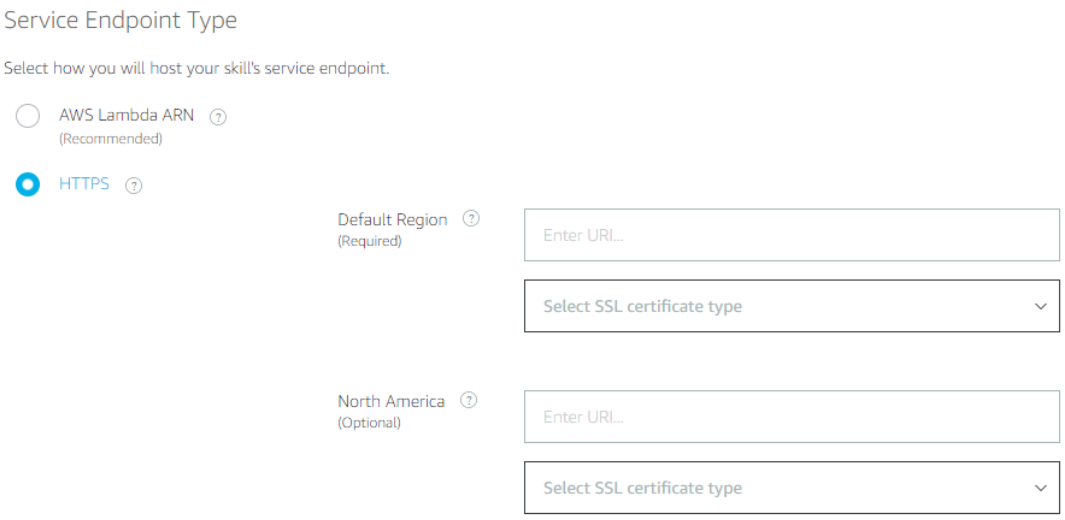

Figura 4.9: Endpoint.

5 Desarrollo de la solución

El objetivo que se plantea en el proyecto es el desarrollo de una solución a través de la cual, haciendo uso de asistentes de voz virtuales como el ofrecido por Amazon mediante su plataforma Alexa, se pueda homogeneizar el acceso a la información disponible a través de infraestructuras IoT. Se particulariza el desarrollo para el caso de uso de la búsqueda de aparcamiento entorno a una calle asociada a una ruta. En esta sección se va a realizar un desarrollo completo, pasando por una solución inicial en que se plantea una búsqueda entorno a un punto hasta llegar a la solución final buscada.

5.1 Arquitectura de alto nivel

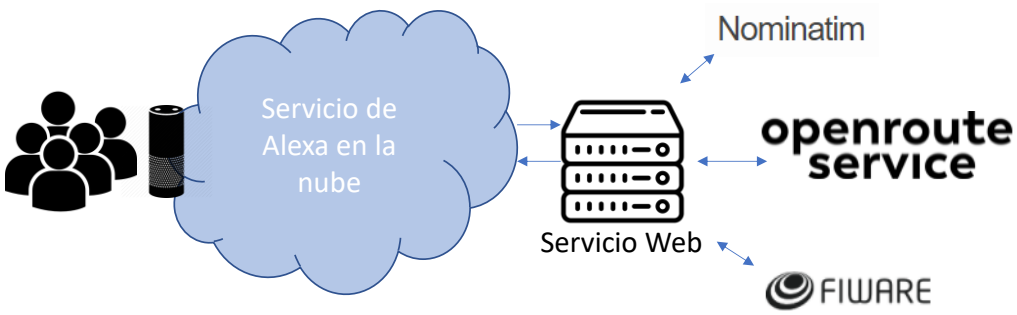

Figura 5.1: Arquitectura de la solución.

En la Figura 5.1 se puede observar el servicio web donde se encuentra el back-end de la skill. Cuando le llegan las peticiones mapeadas a un *intent* y los valores de los *slots* correspondientes, este servicio realiza las peticiones a la plataforma FIWARE, según corresponda. Antes de realizar esta petición es necesario adecuar los datos contenidos en los *slots* para que se adapten al formato de la plataforma. Para ello se hace uso de servicios adicionales como Nominatim o OpenRoutService, por ejemplo, que permiten trasladar localizaciones basadas en nombres de calles a coordenadas y viceversa.

La plataforma FIWARE al recibir la petición geolocalizada realiza la búsqueda dicha información en su base de datos. En esta base de datos se encontrarán los parámetros medidos por el sensor que cumpla con la ubicación geográfica. A continuación, retorna al servicio web la información solicitada.

El servicio web recoge la información proporcionada por FIWARE y realiza las operaciones necesarias antes de proporcionar a Alexa la respuesta que debe dar al usuario.

5.2 Localización y posicionamiento

Antes de proceder con el desarrollo de la skill, tal y como se ha introducido, existe la necesidad de adecuar los datos de proporcionados por el usuario a la representación utilizada en FIWARE. La plataforma solo entiende de coordenadas geográficas, por lo que para facilitar la interacción con el usuario independientemente de si se busca

aparcamiento entorno a un punto o a una calle, es necesario de un medio que permita traducir calles, interfaz que empleará el usuario, a coordenadas.

Antes de profundizar en los servicios empleados para adecuar los datos, se procede a describir el formato de los atributos geoespaciales que se deben enviar en la petición que se remita a la plataforma FIWARE.

Una petición geoespacial permite solicitar a la plataforma un conjunto de sensores y sus métricas en función de su localización. Este tipo de petición requiere por tanto la definición de una serie de atributos como "georel" o "geometry":

- "geometry" define la geometría de referencia desde la que se realiza la búsqueda de la información. Dicho atributo puede hacer referencia a un punto, una línea, un polígono o una caja, los cuales se definen de la siguiente forma:
    - Point: define un punto formado por un par de valores de latitud-longitud, separados por una coma.
    - Line: define una línea mediante una matriz de pares de valores de latitud-longitud, debiendo haber al menos dos pares.
    - Polygon: define un polígono formado por al menos cuatro pares de coordenadas, siendo el último idéntico al primero (ya que un polígono tiene un mínimo de tres puntos reales). Los pares de coordenadas deben estar correctamente ordenados para que los segmentos de línea que componen el polígono permanezcan en el borde exterior del área definida.
    - Box: define una caja representada por una matriz de dos longitudes de pares de latitud-longitud. El primer par es la esquina inferior, el segundo es la esquina superior.
- "georel" tiene por objeto especificar una relación espacial entre las entidades que coincidan con la búsqueda de la petición y la forma de referencia. Dentro de los valores que puede tomar este atributo se puede encontrar los siguientes:
    - Near: las entidades coincidentes deben estar situadas a una determinada distancia de umbral respecto a la geometría de referencia. Se definen unos modificadores para indicar los umbrales de distancia: "maxDistance" y "minDistance", que expresan, en metros, la distancia máxima y mínima, respectivamente, a la que deben situarse las entidades.
    - CoveredBy: hace referencia a todas aquellas entidades que se encuentran en su totalidad dentro de la geometría de referencia. Al resolver una consulta de este tipo, el borde de la forma debe considerarse parte de la misma.
    - Intersects: permite seleccionar las entidades que se encuentran en la zona que cruza con la geometría de referencia.
    - Equals: la geometría asociada a la posición de las entidades coincidentes y a la geometría de referencia debe ser exactamente la misma.

A modo de ejemplo en el caso de que se quisieran localizar las entidades dentro de un área circular con centro en unas determinadas coordenadas y con un radio fijo de 1000 metros, se debería incluir en la consulta la estructura de la Figura 5.2.

```
georel=near;maxDistance:1000&geometry=point&coords=-40.4,-3.5
```

Figura 5.2: Ejemplo con atributos geoespaciales.

Comprendido el formato de las consultas geoespaciales soportadas en FIWARE, se explican a continuación los servicios de los que se han hecho uso para adecuar los nombres de las calles a sus coordenadas. Existen gran variedad de opciones, pero en el ámbito de este proyecto se han empleado OpenRouteService [19] y Nominatim [20].

Ambos son herramientas que permiten hacer peticiones a servicios de geo-codificación, es decir, servicios permiten transformar una descripción de un lugar, como un nombre, una dirección de calle o un código postal, en unas coordenadas geográficas. También implementan el servicio inverso, es decir, a partir de las coordenadas obtener una descripción de las mismas.

A título ejemplificativo de la funcionalidad, supongamos que se pasara la dirección "Paseo de Julio Hauzeur 1, Torrelavega" a cualquiera de los dos servicios. Las respuestas obtenidas en cada caso se muestran en la Figura 5.3 y la Figura 5.4.


```
"type": "Feature",
"geometry": {
    "type": "Point",
    "coordinates": [
        -4.055757,
        43.350117
    ]
},
"properties": {
    "id": "way/602709765",
    "gid": "openstreetmap:address:way/602709765",
    "layer": "address",
    "source": "openstreetmap",
    "source_id": "way/602709765",
    "name": "Paseo de Julio Hauzeur 1",
    "housenumber": "1",
    "street": "Paseo de Julio Hauzeur",
    "postalcode": "39300",
```


Figura 5.3: Respuesta de OpenRouteService.


```
"place_id": 214517178,
"licence": "Data © OpenStreetMap contribut
"osm_type": "way",
"osm_id": 602709765,
"boundingbox": [
    "43.3500647",
    "43.3501754",
    "-4.055823",
    "-4.0556886"
],
"lat": "43.35011405",
"lon": "-4.055757654459002",
"display_name": "1, Paseo de Julio Hauzeur,
```


Figura 5.4: Respuesta de Nominatim.

De lo anterior se observa que ambos servicios ofrecen respuestas similares, las cuales incluyen las coordenadas geográficas, así como el nombre de la calle exacto.

Sin embargo, en nuestro caso se ha optado por emplear principalmente OpenRouteService porque ofrece un mayor número de servicios. Algunos de estos servicios son, por ejemplo, representar el alcance, dibujando un polígono en la zona a la que se puede llegar en un determinado tiempo o distancia, calcular las distancias y los tiempos de varias rutas hasta encontrar la óptima o representar geometrías de puntos o líneas en diferentes formatos y recibir la versión tridimensional en milisegundos, entre otros.

No obstante, el otro servicio de interés para el proyecto además del basado en direcciones, establece rutas de viaje e información de navegación en función de una serie de criterios, como las restricciones de la carretera o las dimensiones del vehículo. Se puede utilizar para coches, camiones, diferentes perfiles de bicicleta, o a pie. Cada uno de estos modos emplea una red de calles cuidadosamente compilada para adaptarse a los requisitos de los perfiles.

Nuevamente a modo de ejemplo se busca generar una ruta para coche a partir de una coordenada origen y otra destino. Supóngase que se indican las calles “Calle Lope de Vega 7, Santander” y “Calle Lope de Vega 1, Santander”. La respuesta devuelta por el servicio OpenRouteService se muestra en la Figura 5.5 e incluye las coordenadas intermedias entre el punto origen y el punto destino. De forma visual, la Figura 5.6 representa el recorrido entre los dos puntos sobre un mapa.


```
"coordinates": [
    [
        -3.799712,
        43.463622
    ],
    [
        -3.799659,
        43.463345
    ],
    [
        -3.799669,
        43.46329
    ],
    [
        -3.799636,
        43.463138
    ]
],
```


Figura 5.5: Respuesta de la ruta en formato JSON.

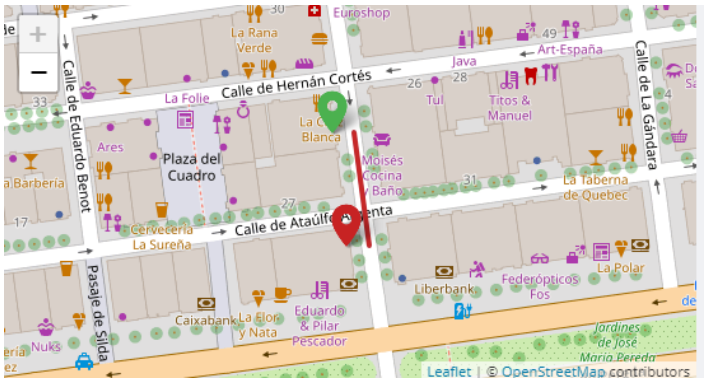

Figura 5.6: Respuesta de la ruta en formato gráfico.

5.3 Lógica de la skill

En este apartado se va a continuar desarrollando el sistema diseñado en la Figura 5.1, ampliando los conocimientos de la lógica de la skill tanto para una primera funcionalidad que circunscribe la búsqueda de aparcamiento entorno a un punto, así como una segunda, basada en el entorno a una calle. En el capítulo 4 se planteó un ejemplo partiendo de un servicio ya definido. En los próximos apartados se profundizará sobre el desarrollo y la configuración de la interfaz de acceso al servicio web que aloja la skill.

5.3.1 Interfaz de acceso al servicio web

A la hora de definir la skill en la que su lógica está desplegada en un entorno bajo el control y gestión del desarrollador y que se exporta como un servicio web es necesario determinar su dirección. Al estar el servidor ejecutándose en local en el equipo de desarrollo, para exportar una URL válida de forma pública se emplea la herramienta Ngrok. Ngrok [21] es capaz de generar un túnel TCP desde Internet a un puerto del servidor en local y vincular dinámicamente una URL al servicio local. Además, incluye otras funciones que han sido útiles durante el desarrollo de la skill como capturar y mostrar todo el tráfico que circula a través del túnel, de forma que se han identificado errores producidos durante la conexión y las peticiones HTTP.

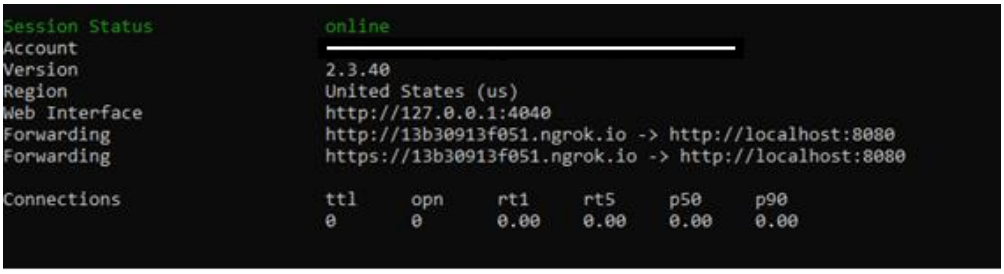

Figura 5.7: Interfaz de Ngrok.

En la Figura 5.7, se puede ver quel la skill se ejecuta en el puerto 8080 del equipo aunque se puede asignar el que se desee.

El servicio web se ha desarrollado empleando el framework de NodeJS Express [22]. NodeJS [23] proporciona a los desarrolladores todo tipo de mecanismos para crear

aplicaciones y servicios en JavaScript. Ofrece un entorno de código abierto y multiplataforma que trabaja en tiempo de ejecución.

Desde el punto de vista de un desarrollador de servicios web, NodeJS presenta un gran número de ventajas como un alto rendimiento y una gran capacidad de escalabilidad. Además de una serie de ventajas derivadas del lenguaje JavaScript, como la compatibilidad con distintos sistemas operativos u otros lenguajes de programación. Cabe mencionar que cuenta con una comunidad de desarrolladores muy activa, gracias a la cual se proporciona acceso a una gran variedad de paquetes reutilizables.

Para el desarrollo de servicios web, Express es uno de los frameworks más usados de NodeJS y proporciona una serie mecanismos para la gestión de peticiones HTTP en diferentes URL.

```
1  const express = require('express');
2  const { ExpressAdapter } = require('ask-sdk-express-adapter');
3
4  const app = express();
5  const skillBuilder = require('./skillBuilder.js').skillBuilder;
6  const skill = skillBuilder.create();
7  const adapter = new ExpressAdapter(skill, true, true);
8
9  app.post('/', adapter.getRequestHandlers());
10 app.listen(8080, () => {
11     console.log( 'API REST corriendo en http.//localhost:8080')
12
13 });
```

Figura 5.8: Código de servidor web.

A la hora de implementar el servidor que aloje la lógica de la skill, hay que incluir funcionalidades que verifiquen la procedencia de las peticiones y validen que proveen del entorno de Alexa. De esta manera se evitan ciertos problemas de seguridad asociados. El SDK de Alexa proporciona una librería, que apoyándose en Express, permite verificar el origen y el momento de recepción de las peticiones. En la Figura 5.8, se muestra cómo se incluye el paquete ask-sdk-express-adapter que aporta estas funcionalidades.

A la hora de generar el soporte web para la skill se hace uso de las funciones de la librería ExpressAdapter, la cual antes de derivar la gestión de la petición a los manejadores específicos, filtra el contenido recibido.

5.3.2 SkillBuilder

El archivo skillBuilder.js contiene el código de la skill. En este apartado se realiza un análisis de los aspectos más importantes de su contenido.

```
1  const Alexa = require('ask-sdk-core');
2  const axios = require('axios').default;
```

Figura 5.9: Módulos importados.

En la Figura 5.9 se muestran los módulos importados que se utilizarán posteriormente. El primero el ask-sdk-core hace referencia al ASK SDK v2 para NodeJS, un kit de desarrollo de skills de Alexa que facilita su implementación. El segundo por su parte,

Axios [24] es fundamental para completar las funciones de la skill que se desarrolla en este trabajo, pues permite hacer distintas peticiones como cliente HTTP. No obstante, se pueden utilizar otras librerías para realizar peticiones HTTP como GOT [25] o SuperAgent [26].

Continuando con el análisis del código, se implementa una serie de *handler* o manejadores que son los encargados de ejecutar la lógica de un *intent* determinado. Por tanto, se encuentrán uno por cada *intent* presente en la skill, incluyendo los predefinidos por Alexa.

El *handler* de la Figura 5.10, hace referncia a la invocacion de nuestra skill.

```
const LaunchRequestHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
    },
    handle(handlerInput) {
        const speakOutput = 'Bienvenido a BUSCA APARCAMIENTO. Dígame si quiere obtener el\
        número de plazas de aparcamiento de una calle en un radio de 50 metros\
        o si quiere buscar aparcamiento dentro de una ruta.';

        return handlerInput.responseBuilder
            .speak(speakOutput)
            .reprompt(speakOutput)
            .getResponse();
    }
};
```

Figura 5.10: Handler de bienvenida.

De forma general, un *handler* consta de dos partes relacionas con el *intent* al que se llama:

- *CanHandle*: se encarga de identificar si el *handler* debe ejecutarse antes una petición específica. El código mostrado en la Figura 5.10 es se asocia al tipo de petición LaunchRequest, remitida para el lanzamiento de la skill. El resto de las solicitudes incluyen el tipo IntentRequest.
- *handle*: si el *handler* se asocia a la petición (*canHandle* retorna un true), esta función incluye el código asociado al *intent* correspondiente. Así en el ejemplo de la Figura 5.10, el código devuelve el mensaje de bienvenida que Alexa Voice Service transmitirá al usuario.

A continuación, se explicará los dos *handler*s contenidos en nuestra skill. Por una parte, un *handler* inicial en el que se busca el número de aparcamientos entorno a un punto dado por las coordenadas de una calle. Y el segundo, partiendo de la solución inicial, busca la ubicación de las plazas de aparcamiento dentro de una calle entendida como una polilínea. Estos *handlers* están asociados a distintos *intent*s (“GetParkingPointFWIntent” y “GetParkingPointFWIntent”), los cuáles se explicarán en apartados posteriores cuando se traten sus respectivos modelos de interacción.

5.3.2.1 Aparcamientos disponibles en torno a un punto

En un principio para facilitar el desarrollo de la solución final, se decidió plantear una solución más sencilla que basara su funcionamiento en la búsqueda de aparcamiento entorno a un punto, como se muestra en la Figura 5.11.

Figura 5.11: Búsqueda de aparcamiento entorno a un punto.

Para realizar esta búsqueda se ha aprovechado la información del modelo de datos de estacionamiento de FIWARE. Vamos a ver qué tipo de información proporciona este tipo de modelo de datos a través de un ejemplo, Figura 5.12.


```
{
  "id": "santander:daoiz_velarde_1_5:3",
  "type": "ParkingSpot",
  "status": {
    "value": "free",
    "metadata": {
      "timestamp": {
        "type": "DateTime",
        "value": "2018-09-21T12:00:00"
      },
      "parkingPermit": {
        "type": "Property",
        "value": "yes"
      }
    }
  },
  "category": {
    "value": ["onStreet"]
  },
  "refParkingSite": {
    "type": "Relationship",
    "object": "santander:daoiz_velarde_1_5"
  },
  "name": {
    "value": "A-13"
  },
  "location": {
    "type": "geo:json",
    "value": {
      "type": "Point",
      "coordinates": [-3.80356167695194, 43.46296641666926]
    }
  }
}
```


Figura 5.12: Modelo de datos de estacionamiento.

Al igual que resto de modelos de datos, comienzan con el identificador único de la entidad, seguido del tipo y la última fecha de actualización. Este tipo de entidad cuenta con una serie de parámetros que indican, por ejemplo, el tipo de aparcamiento (a pie de calle o subterráneo), la disponibilidad (libre, ocupado, abierto o cerrado), la referencia del

propio sensor, así como la ubicación del mismo. Sin embargo, nos centramos únicamente en el estado y la posición del sensor.

Una vez conocido el modelo de datos, estamos en disposición de realizar una petición POST a la plataforma FIWARE, cuyo cuerpo se muestra en la Figura 5.13.

```
data: {
    "entities": [
        {
          "idPattern": ".*",
          "type": "ParkingSpot"
        }
      ],
        "attributes": [
          "status",
          "location"
      ],
       "expression": {
            "q": "status==free",
            "georel": "near;maxDistance:50",
            "geometry": "point",
            "coords": `${lat},${lon}`
          },
       "metadata": [
          "accuracy",
          "timestamp"
      ]
},
```

Figura 5.13. Cuerpo de la petición POST entorno a un punto.

En el cuerpo de la petición se puede observar que nos interesan todas las entidades de tipo “ParkingSpot”, y dentro de éstas, sus atributos de estado y localización. En el apartado 5.2 se explicó cómo se definían las consultas geoespaciales. De forma general, esta consulta busca las entidades libres (“status == free”), en una distancia no mayor a 50 m (“georel: near; maxDistance:50”) desde una posición que se pasan como variables “lat” y “lon”. Estos valores se corresponden con la respuesta de uno de los servicios ya explicados, Nominatim o OpenRouteService.

En la Figura 5.14 se muestra la respuesta que devolvería la plataforma FIWARE, para el caso que la calle de entrada en el modelo de interacción fuera “Calle Hernán Cortes 28, Santander”.

Figura 5.14: Respuesta petición POST entorno a un punto.

En dicha figura se puede observar que la respuesta devuelve un array con los sensores disponibles. Para saber el número de aparcamientos de la zona, únicamente es necesario contar el número de elementos, de tal forma que Alexa puede responder como se muestra en la Figura 5.15, donde "n" es la longitud y "centro" la calle pedida por el usuario.

```
let speechText = `Hay ${n} plaza/s de aparcamiento en la ${centro}.`;
```

Figura 5.15: Respuesta proporcionada por Alexa.

5.3.2.2 Aparcamientos disponibles en torno a una calle

Lograda la funcionalidad anterior con éxito, se planteó como conseguir la calle completa como una polilínea, formada por todas las coordenadas geográficas generadas a partir de un punto origen y un punto destino mediante el servicio OpenRouteService.

Inicialmente se planteó la idea de por cada coordenada contenida en la ruta realizar un ciclo que la recorriera realizando círculos como se muestra en la Figura 5.16, pero tenía como inconveniente que los puntos dentro la ruta no tenían una distancia uniforme, por lo que, al definir la búsqueda con un radio fijo, no se consideraba todo el área. Por este motivo se descartó esta idea.

Figura 5.16: Aparcamiento entorno a una ruta formada por varios puntos.

Finalmente, partiendo de la ruta anterior se planteó la solución final en la que se devuelve el contenido de un polígono formado a través de todas las coordenadas que forman la polilínea. La Figura 5.17 muestra gráficamente este planteamiento.

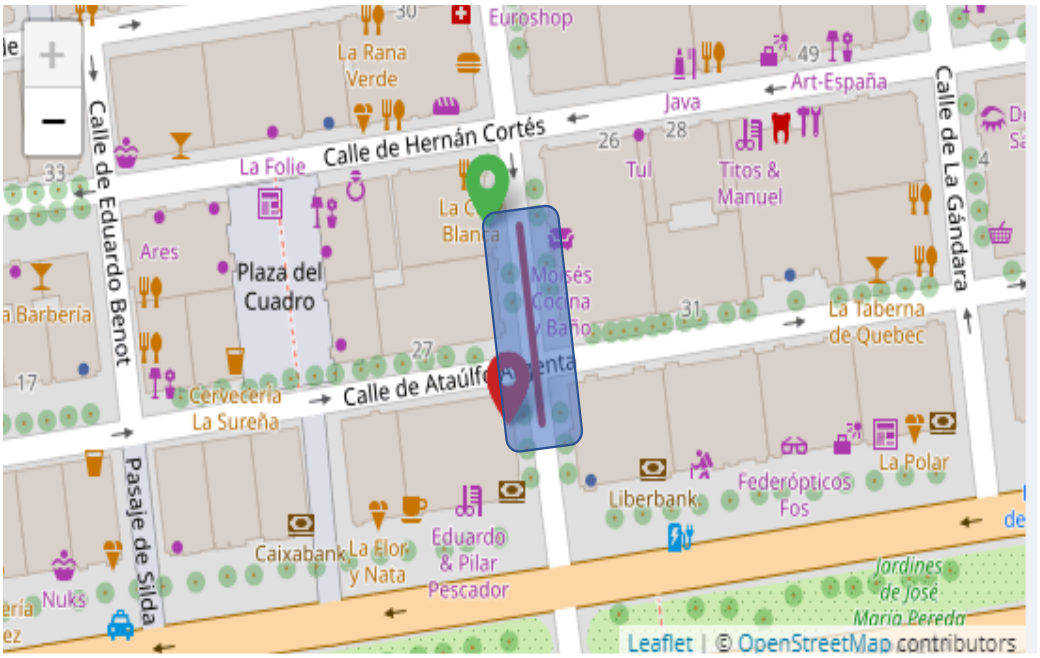

Figura 5.17: Aparcamiento dentro de la ruta formada por un polígono.

Partiendo de la petición POST anterior, se modificaron los parámetros geoespaciales, como se muestra en la Figura 5.18.

```
data: {
    "entities": [
        {
            "idPattern": ".*",
            "type": "ParkingSpot"
        }
    ],
      "attributes": [
        "status",
        "location"
    ],
    "expression": {
        "q": "status==free",
        "georel": "coveredBy",
        "geometry": "polygon",
        "coords": `${polygon}`
    },
    "metadata": [
      "accuracy",
      "timestamp"
    ]
},
```

Figura 5.18: Cuerpo de la petición POST entorno a una calle.

A diferencia de la primera petición, ésta utiliza un polígono como geometría de referencia. Sin embargo, la ruta que hemos obtenido de OpenRouteService está formada por un conjunto de puntos entendidos como una línea. Por tanto, es necesario dotar a estos puntos de un área y mediante la opción “CoveredBy” buscar dentro de ellos.

Para ello se hace uso de los servicios de la librería TURF [27]. Esta librería incluye operaciones espaciales tradicionales, funciones auxiliares para crear datos con formato GeoJSON y herramientas de clasificación de datos y estadísticas. Gracias a TURF se puede llevar a cabo tareas como operaciones estadísticas dentro de un conjunto de puntos, determinar distancias, construir objetos geográficos o transformarlos en otro tipo de objetos.

Para nuestro caso, se busca transformar una ruta lineal formada por puntos, en un área o polígono que la circunscriba para poder detectar los sensores de aparcamiento en todo el rango de la calle. Para ellos se emplea la función *buffer* [28], en la cual es necesario especificar las coordenadas de la ruta siguiendo el estándar WGS84 (latitud, longitud), el radio, y las unidades de este último como se muestra en la Figura 5.19.

```
var ruta = turf.line([Conjunto de Coordenadas]);
var polígono = turf.buffer(ruta, 20, {units: 'meters'});
```

Figura 5.19: Ejemplo de uso de la función buffer.

Una vez generada la forma buscada, se pasa en la petición dentro de la variable “Polygon”.

En la Figura 5.20 se muestran los aparcamientos disponibles entre dos puntos, origen la “Calle Hernán Cortés 28, Santander” y destino la “Calle Hernán Cortés 32, Santander”.

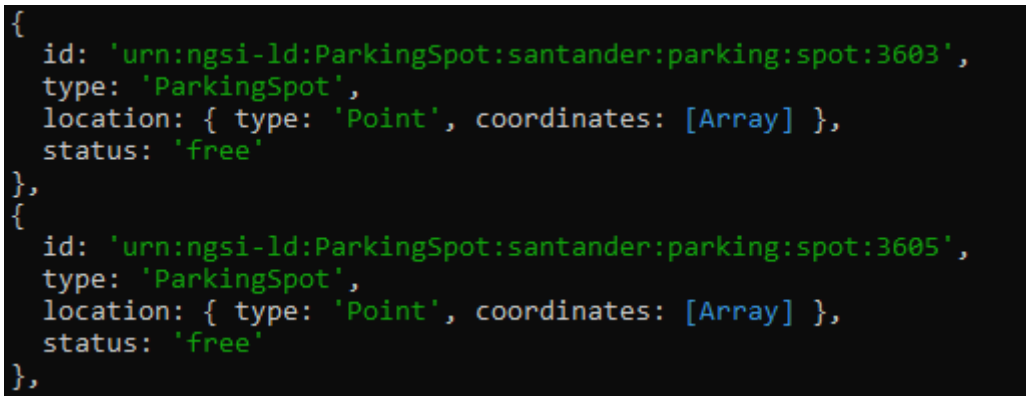

Figura 5.20: Respuesta petición POST.

En este caso, no se busca devolver el número de aparcamientos disponibles, aunque se podría replicando la operativa anteriormente descrita, sino la ubicación de estos. Para ello se hace uso del servicio de geo-codificación inverso de forma que se traduce la posición del sensor a una calle y número, de tal forma que Alexa pueda responder al usuario empleando un lenguaje amigable. La Figura 5.21 muestra el texto retornado, donde la variable "name" recoge la traducción de coordenadas a nombre de calle.

```
let speechText = `Hay una plaza de aparcamiento en ${name}.`;
```

Figura 5.21: Respuesta proporcionada por Alexa.

5.4 Modelo de interacción

En esta sección se va a desarrollar brevemente los modelos de interacción para cada funcionalidad previamente explicada. Ambas funciones están contenidas bajo la misma skill, "Busca Aparcamiento".

Para poder invocar esta skill y acceder a su contenido se ha definido un *invocation name* que coincide con el propio nombre de la skill "busca aparcamiento". Para cada funcionalidad se ha definido un *intent* que llama a cada acción con sus correspondientes *utterances* y *slots*. En los próximos subapartados se explicarán cada i*ntent* por separado.

5.4.1 Aparcamientos disponibles en torno a un punto

El *intent* mostrado en la Figura 5.22 representa la acción de búsqueda del número de plazas de aparcamiento disponibles entorno a un punto.

Figura 5.22: GetParkingPointFWIntent.

En esta figura se pueden observar las *utterances* para llamar al *intent* creado, “Dime el número de aparcamiento en la {centro}” y “numero de aparcamiento en la {centro}”. Dentro de estas *utterances* se encuentra entre corchetes el *slot* denominado “centro”, el cual recoge la calle que se toma como punto medio. Se ha definido como un *slot* de tipo AMAZON.SearchQuery, que representa una cadena de caracteres libres. En el lenguaje español no hay aún definido un tipo dirección completa, si bien sí existe un tipo calle AMAZON.StreetName, pero no incorpora el número. En un principio se planteó utilizar este tipo de *slot* seguido de un *slot* de tipo AMAZON.Number, pero su comportamiento no era el adecuado al no ser capaz de reconocer los campos por separado.

5.4.2 Aparcamientos disponibles en torno a una calle

El *intent* mostrado en la Figura 5.23 representa la acción de busca la ubicación de plazas de aparcamiento disponibles en torno a una calle.

Figura 5.23: GetParkingRouteFWIntent.

En esta figura se pueden observar la *utterances* para llamar al *intent* creado, “Busca aparcamiento dentro de una ruta”. A diferencia del *intent* anterior, la *utterances* no contiene ningún *slot* ya que con el fin de facilitar la interacción con la skill, se ha definido un diálogo en el que Alexa va pidiendo dichos valores. Aun así, es necesario definir los *slots* mostrados en la Figura 5.24 para tratarlos más tarde.

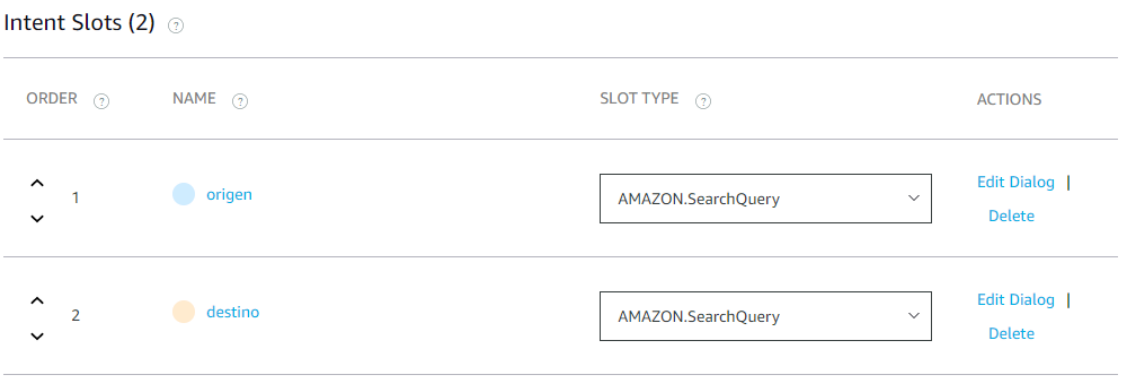

Figura 5.24: Slots Origen y Destino.

Una vez creados los slots, se define un diálogo para que Alexa los pida al usuario de manera obligatoria. Esto lleva acabo desde la opción “Edit Dialog” del menú de configuración del *slot*.

Tal y como se observa en la Figura 5.25 para el caso del *slot* "origen", se define como se quiere que Alexa solicite el *slot*. Puesto que en el *intent* no se ha definido que se capture el valor del slot, pero sí se ha definido este como obligatorio, tras llamar al *intent* continuara pidiendo una calle de origen y al igual que en los *intents* se pueden definir todas las posibles formas en la que le usuario puede responder. En este caso, Alexa diría "Dime una calle de origen", y el usuario debería responder directamente con la dirección de la calle completa. Seguidamente, y fijado el valor del *slot* "origen", se sigue un proceso similar para el *slot* "destino". Así, Alexa pedirá una calle destino, diciendo "Dime una calle de destino", y el usuario responderá directamente con la calle completa.

Figura 5.25. Editor de diálogos.

5.5 Evaluación

Estudiadas todas las características de la skill "Busca Aparcamiento", es el momento de comprobar el funcionamiento de la misma. A continuación, se ilustra en la Figura 5.26 el diálogo que un usuario cualquiera mantiene con Alexa para acceder a las funcionalidades que se han desarrollado.

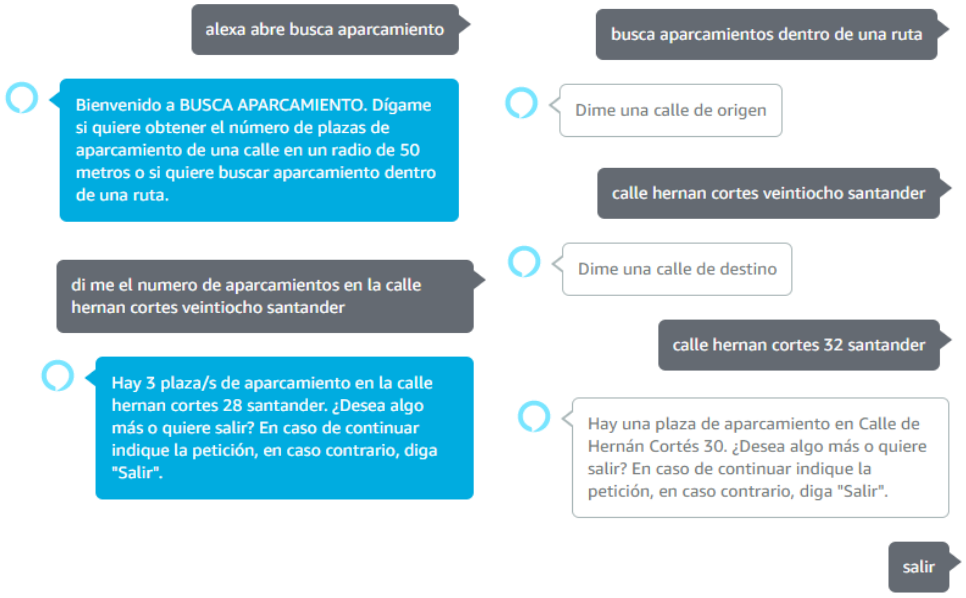

Figura 5.26: Interacción con Busca Aparcamiento

El diálogo comienza lanzando la skill a través del *invocation name* “busca aparcamiento”. Es entonces cuando Alexa reconoce que se trata de una petición LaunchRequest y responde con el mensaje de bienvenida asociado a la skill solicitada.

El mensaje de bienvenida indica las dos funciones que están disponibles. Para acceder a la funcionalidad que devuelve el número de plazas se utiliza la *utterance* correspondiente, en este caso, “dime el número de aparcamientos en la calle Hernán Cortés 28, Santander”. Una vez mapeada al *intent* correspondiente, Alexa se queda con el valor de calle para pasárselo al servicio desarrollado. En el servicio se traduce el nombre de la calle a coordenadas y se realiza la petición a FIWARE. Tras recibir la respuesta, se adecua para que el usuario pueda comprenderla.

Finalizado el ciclo de una petición, la skill desarrollada se mantiene a la espera de otra petición o bien del cierre de sesión.

Respecto a la funcionalidad que busca la ubicación de la plaza de aparcamiento en una calle asociada a una ruta, se utiliza como *utterance* “busca aparcamiento dentro una ruta”, Es entonces cuando comienza un diálogo para pedir los valores para generar la ruta. Al igual que en caso anterior, se asocia a un *intent* propio y se pasan los valores al servicio, para más tarde realizar la petición a FIWARE y obtener la respuesta mostrada en la Figura 5.26.

6 Conclusiones y líneas futuras

El propósito inicial planteado para el proyecto ha sido dotar a los usuarios de una utilidad que facilite la búsqueda de aparcamiento en una ciudad empleando el lenguaje natural. Para ello, se ha desarrollado una skill de Alexa que aprovecha la información que recogen las plataformas de IoT de una Smart City. A continuación, se va a realizar una valoración y balance de lo que ha significado este proyecto además de recoger una serie de conclusiones que se han extraído durante el desarrollo del trabajo, así como posibles ampliaciones y líneas futuras.

6.1 Conclusiones

El proyecto se ha basado en la utilización de la información recogida en una ciudad inteligente y que ésta hace accesible a través de plataformas de IoT. Esta información se ha integrado en una solución que empleando el asistente virtual remite al usuario información de la situación de los aparcamientos de la ciudad de Santander.

El trabajo se ha iniciado con un análisis tanto de soluciones y plataformas empleadas más habitualmente en el entorno de las ciudades inteligentes, para posteriormente centrarse en la ofrecida por FIWARE, pues es una de las habilitadas en la ciudad de Santander. Adicionalmente, se ha realizado un estudio de diferentes asistentes virtuales, descubriendo las bondades que ofrecen como nuevas metodologías de interacción con los usuarios. Estos abren un escenario nuevo y lleno de oportunidades, al dotar de una experiencia totalmente inmersiva, con una interacción natural y sin fricciones entre hombre y máquina.

Gran parte del proyecto ha girado en torno a distintos servicios, ya sea al servicio creado que se comunica con el servicio de Alexa en la nube o a los distintos servicios que se han empleado para facilitar el acceso a la información disponible en la plataforma FIWARE por parte del usuario. En este último caso, dado que las peticiones realizadas a esta plataforma basan su funcionamiento en el uso de coordenadas geográficas, se han utilizado distintos servicios web o API para traducir nombres de calles a coordenadas y viceversa. Gracias a ello ha sido posible indicar en qué posición se encuentra una plaza de aparcamiento libre de manera que el usuario sea capaz de entenderlo. Aparte de este servicio, se ha hecho uso de otros servicios como por ejemplo la generación de una ruta. Mediante la combinación de ellos se ha dado lugar a un servicio nuevo más potente que cada uno de los anteriores por separado.

Una vez que se comprende el funcionamiento API o de servicios web, las posibilidades son muy amplias para desarrollar tu aplicación, incorporando otros servicios ya existentes. Además, cuentan con una serie de ventajas que facilitan las tareas del desarrollador, por ejemplo, la interacción entre el proveedor y el desarrollador que solicita el servicio es completamente independiente de la plataforma y el lenguaje.

Durante el desarrollo del proyecto, aun considerando el resultado exitoso, han surgido distintas dificultades. Una vez establecida la temática, los problemas que surgieron fueron más de carácter técnico. Siendo la primera vez que me enfrentaba al desarrollo de una skill, fue necesario un tiempo para comprender tanto el funcionamiento como el entorno

de trabajo. Previamente había utilizado otros lenguajes de programación, pero nunca JavaScript. No obstante, no he encontrado especial dificultad en su aprendizaje y resulta muy intuitivo una vez se tiene una base de programación. Además, existen disponibles una amplia variedad de tutoriales y manuales.

Por otra parte, este proyecto me ha permitido afianzar los conceptos de asignaturas relacionados con redes y servicios que he recibido durante la carrera, pudiendo tener así, una visión más completa del sistema desarrollado.

6.2 Líneas futuras

Se pueden realizar diversas mejoras sobre la aplicación presentada, en aras de lograr una interacción aún más amigable. Para ello se podrían incorporar algunas de las opciones que soportan el desarrollo de skills, como, por ejemplo, Alexa Presentation Language (APL) mediante la cual se pueden crear experiencias visuales para aquellos dispositivos que además de soportar Alexa tengan pantalla.

Actualmente la skill devuelve un solo aparcamiento de los que encuentran disponibles, pero en realidad la respuesta obtenida de la plataforma de gestión incluye el conjunto de todas las coordenadas de los sensores sobre los que no hay un coche estacionado. Sería adecuado poner a disposición del usuario todas estas ubicaciones y que sea él/ella quien decida cual le conviene más. En este sentido, y empleando APL, se podría incorporar un mapa en el que se representara cada punto. Otra alternativa para dotar al servicio de una interfaz gráfica, aunque quizás más complicada, sería desarrollar una aplicación móvil que contenga el servicio desarrollado, y que Alexa sea capaz de interactuar con dicha aplicación al igual que hace con Spotify cuando le pedimos que ponga una canción.

Otra opción interesante de incorporar y también presente entre las características de Alexa es el uso de Proactive Events API, con la que se pueden enviar eventos a Alexa. Los eventos representan datos factuales que podrían interesar a un cliente. Al recibir un evento, Alexa entrega proactivamente la información a los clientes que han elegido recibir estos eventos. Un evento podría notificar una plaza de aparcamiento que queda libre en una zona determinada.

Continuando en la línea de la aplicación desarrollada y con uno de los objetivos planteados al principio del proyecto como era reducir el riesgo accidentes en carretera debido a distracciones causadas por tecnologías que interfieren en la conducción, Amazon ha desarrollado un dispositivo de la serie Echo destinada para vehículos, el Amazon Echo Auto. Gracias a este dispositivo se puede incorporar el asistente a cualquier vehículo, una vez conectado se puede interactuar a través de los múltiples micrófonos que dispone y Alexa respondería a través de los altavoces del vehículo. Tras un estudio superficial del dispositivo, se ha observado que una de las características con las que cuenta es la capacidad de conectarse a la aplicación móvil de Alexa, en la que se encuentran instaladas las skills. Sería interesante continuar el estudio previo para utilizar la skill desarrollada desde el Echo Auto.

En cuanto a futuras líneas de trabajo más generales, se podrían desarrollar otros servicios sobre la información que se tiene de la ciudad. Se ha mencionado que la plataforma FIWARE cuenta con una amplia variedad modelos de datos estructurados de forma homogénea y estandarizada, por lo que se puede acceder a ellos de manera relativamente sencilla. Por ejemplo, partiendo las peticiones geolocalizadas, se puede

desarrollar una aplicación que proporcione información meteorológica de distintas partes de la ciudad. De esta forma, se podría poner una gran cantidad de datos al servicio de la población.

Además, si se tienen en cuenta los avances tecnológicos en el ámbito del Big Data y el Machine Learning, los asistentes virtuales cada vez podrán contar con más información, por lo que pueden convertirse en sistemas altamente inteligentes y sustituir a las interfaces tradicionales basadas en pantallas o botones.

Bibliografía

[1] Red, G. T. (2021). *Ciudades Inteligentes • ESMARTCITY*. Obtenido de https://www.esmartcity.es/ciudades-inteligentes.

[2] FIWARE. (2021). *The Open Source platform for our smart digital future - FIWARE*. Obtenido de https://www.fiware.org/.

[3] Vila, K., & Celso Fernández, A. (2021). *Interfaz de Acceso en Lenguaje Natural a la Información de la Universalización de la Enseñanza Superior*. Obtenido de https://www.researchgate.net/publication/237082026_Interfaz_de_Acceso_en_Lenguaje_Natural_a_la_Informacion_de_la_Universalizacion_de_la_Ensenanza_Superior.

[4] Digital, M. d. (2021). *Plan de Tecnologías del Lenguaje - Las Tecnologías del Lenguaje*. Obtenido de https://plantl.mineco.gob.es/tecnologias-lenguaje/Paginas/tecnologias-lenguaje.aspx.

[5] ONU. (2018). *Las ciudades seguirán creciendo, sobre todo en los países en desarrollo | ONU DAES | Naciones Unidas Departamento de Asuntos Económicos y Sociales*. Obtenido de https://www.un.org/development/desa/es/news/population/2018-world-urbanization-prospects.html.

[6] *Plan Nacional de Ciudades Inteligentes*. (2015). Obtenido de https://plantl.mineco.gob.es/planes-actuaciones/Bibliotecaciudadesinteligentes/Detalle%20del%20Plan/Plan_Nacional_de_Ciudades_Inteligentes_v2.pdf.

[7] Cárdenas, A. (2016). *¿Qué es una plataforma IoT?* Obtenido de https://secmotic.com/plataforma-iot/.

[8] SynchroniCity. (2020). *A universal approach to developing, procuring and deploying IoT- and AI-enabled services.* Obtenido de https://synchronicity-iot.eu.

[9] OneM2M. (2021). *Sets Standards For The Internet Of Things & M2M*. Obtenido de https://www.onem2m.org/.

[10] FIWARE. (2021). *Step-by-Step for NGSI-v2*. Obtenido de https://fiware-tutorials.readthedocs.io/en/latest/index.html.

[11] FIWARE. (2013). *FI-WARE and Smart Cities in Santander*. Obtenido de https://www.fiware.org/2013/09/19/santander-smart-city-event/.

[12] Nolla, F. C. (s.f.). *CVC. Congreso de Sevilla. La lengua española y las nuevas tecnologías*. Obtenido de https://cvc.cervantes.es/obref/congresos/sevilla/tecnologias/mesaredon_casacuberta.htm.

[13] Amazon. (2021). *Amazon Alexa Official Site: What is Alexa?* Obtenido de https://developer.amazon.com/es-ES/alexa .

[14] Google. (2021). *Google Assistant, your own personal Google*. Obtenido de https://assistant.google.com/.

[15] Apple. (2021). *Siri*. Obtenido de https://www.apple.com/es/siri/.

[16] Microsoft. (2021). *¿Qué es Cortana?* Obtenido de https://support.microsoft.com/es-es/topic/-qu%C3%A9-es-cortana-953e648d-5668-e017-1341-7f26f7d0f825.

[17] Samsung. (2021). *Bixby*. Obtenido de https://www.samsung.com/es/apps/bixby/.

[18] *Ccadb-public.secure.force.com*. (2021). Obtenido de https://ccadb-public.secure.force.com.

[19] Openrouteservice.org. (2021). *Openrouteservice*. Obtenido de https://openrouteservice.org/

[20] Nominatim.org. (2021). *Nominatim*. Obtenido de https://nominatim.org/.

[21] Ngrok. (2021). ngrok - secure introspectable tunnels to localhost. Obtenido de https://ngrok.com/.

[22] Express. (2021). *Infraestructura de aplicaciones web Node.js*. Obtenido de https://expressjs.com/es/.

[23] Mozilla. (2021). *Introducción a Express/Node - Aprende sobre desarrollo web | MDN*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction.

[24] Axios. (2021). *Getting Started | Axios Docs*. Obtenido de https://axios-http.com/docs/intro.

[25] npmjs. (2021). *GOT*. Obtenido de https://www.npmjs.com/package/got.

[26] GitHub. (2021). *SuperAgent — elegant API for AJAX in Node and browsers.* Obtenido de https://visionmedia.github.io/superagent.

[27] Turfjs.org. (2021). *Turf.js | Advanced Geospatial Analysis.* . Obtenido de http://turfjs.org/.

[28] GitHub. (2021). *Turfjs/turf-buffer*. Obtenido de https://github.com/Turfjs/turf-buffer.